

Business Document Naming and Design Rules Version 1.1

Committee Specification Draft 01 /
Public Review Draft 01

11 January 2017

Specification URIs

This version:

<http://docs.oasis-open.org/ubl/Business-Document-NDR/v1.1/csprd01/Business-Document-NDR-v1.1-csprd01.xml> (Authoritative)
<http://docs.oasis-open.org/ubl/Business-Document-NDR/v1.1/csprd01/Business-Document-NDR-v1.1-csprd01.html>
<http://docs.oasis-open.org/ubl/Business-Document-NDR/v1.1/csprd01/Business-Document-NDR-v1.1-csprd01.pdf>

Previous version:

N/A

Latest version:

<http://docs.oasis-open.org/ubl/Business-Document-NDR/v1.1/Business-Document-NDR-v1.1.html>
<http://docs.oasis-open.org/ubl/Business-Document-NDR/v1.1/Business-Document-NDR-v1.1.pdf>

Technical Committee:

OASIS Universal Business Language (UBL) TC

Chair:

G. Ken Holman (gkholman@CraneSoftwrights.com), Crane Softwrights Ltd.

Editors:

Kenneth Bengtsson (kenneth@alfa1lab.com), Individual
Erlend Klakegg Bergheim (erlend.klakegg.bergheim@difi.no), Difi-Agency for Public Management and eGovernment
G. Ken Holman (gkholman@CraneSoftwrights.com), Crane Softwrights Ltd.

Related work:

This specification supercedes:

[**Business-Document-NDR-v1.0**] *Business Document Naming and Design Rules Version 1.0*. Edited by Tim McGrath, Andy Schoka and G. Ken Holman. 18 January 2017. OASIS Standard. <http://docs.oasis-open.org/ubl/Business-Document-NDR/v1.0/os/Business-Document-NDR-v1.0-os.html>. Latest version: <http://docs.oasis-open.org/ubl/Business-Document-NDR/v1.0/Business-Document-NDR-v1.0.html>.

Abstract:

This specification prescribes a set of naming and design rules used to create document model validation artefacts associated with abstract information bundles formally described using the Core Component Technical Specification 2.01 [**CCTS**] in either or both of XML documents using W3C Schema XSD files and OASIS Context/value association files, or/and JSON documents using expressions.

Status:

This document was last revised or approved by the OASIS Universal Business Language TC on the above date. The level of approval is also listed above. Check the “Latest version” location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl#technical.

TC members should send comments on this specification to the TC’s email list. Others should send comments to the TC’s public comment list, after subscribing to it by following the instructions at the “**Send A Comment**” button on the TC’s web page at <https://www.oasis-open.org/committees/ubl/>.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page at <https://www.oasis-open.org/committees/ubl/ipr.php>.

See [Appendix A, \(informative\) Release notes](#) for more information regarding this release package.

Citation format:

When referencing this specification the following citation format should be used:

[Business-Document-NDR-v1.1] *Business Document Naming and Design Rules Version 1.1*. Edited by Kenneth Bengtsson, Erlend Klakegg Bergheim and G. Ken Holman. 11 January 2017. OASIS Committee Specification Draft 01 / Public Review Draft 01. <http://docs.oasis-open.org/ubl/Business-Document-NDR/v1.1/csprd01/Business-Document-NDR-v1.1-csprd01.html>. Latest version: <http://docs.oasis-open.org/ubl/Business-Document-NDR/v1.1/Business-Document-NDR-v1.1.html>.

Notices

Copyright © OASIS Open 2001-2017. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the “OASIS IPR Policy”). The full [Policy](#) may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an “AS IS” basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS’ procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name “OASIS” is a trademark of [OASIS](#), the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <https://www.oasis-open.org/policies-guidelines/trademark> for guidance.

Table of Contents

1	Introduction	6
1.1	Basic concepts	6
1.1.1	Naming and design rules	6
1.1.2	Modeling concepts	8
1.1.3	Business Information Entities	9
1.2	Terminology	9
1.2.1	Key words	9
1.2.2	Terms and Definitions	10
1.2.3	Symbols and Abbreviations	11
1.3	Normative References	11
1.4	Non-Normative References	12
2	Context of use and application of these rules	13
3	Information modeling	14
3.1	Document ABIEs	14
3.2	ABIE library	14
3.3	Extensions	14
4	Dictionary information	16
4.1	Dictionary information overview	16
4.2	Dictionary information values	16
4.3	Abbreviations	17
4.4	Dictionary information for BIEs	18
4.4.1	Component type for BIEs	18
4.4.2	Dictionary information for ABIEs	18
4.4.3	Dictionary information for BBIEs	19
4.4.4	Dictionary information for ASBIEs	23
4.4.5	Dictionary entry names	26
4.5	Structure of an ABIE	26
5	XML validation artefacts	27
5.1	XML validation artefacts overview	27
5.2	XML Namespaces	27
5.3	XSD import/include tree	29
5.4	Schema fragment annotations	30
5.5	XML schema fragments and declarations	31
5.5.1	XML schema fragments for Document ABIEs	31
5.5.2	XML schema fragment for Library ABIEs	31
5.5.3	XML schema fragment for BBIEs	32
5.5.4	XML schema declarations for all BIEs	32
5.5.5	XML schema declarations for ABIEs	33
5.5.6	XML schema declarations for ASBIEs	35
5.5.7	XML schema declarations for BBIEs	35
5.5.8	XML schema declarations for Qualified Data Types	36
5.6	Extension XML schema fragments and declarations	36
5.6.1	Extension information in XML	36
5.6.2	Extension collection schema fragments and declarations	36
5.6.3	XML schema fragment for the extension content data type declaration	38
5.7	Qualified data types XML schema fragment and declarations	39
5.8	Unqualified data types XML schema fragment and declarations	40
5.9	CCTS Core Component Types XML schema	42
5.10	XML attribute names	42
5.11	Data type qualifications in XML	43
6	JSON validation artefacts	46
6.1	JSON validation artefacts overview	46
6.2	JSON preservation of XML namespaces	46
6.3	JSON schema file references	46
6.4	JSON schema fragment annotations	47

6.5 JSON Schema fragments and declarations	48
6.5.1 JSON schema fragments for Document ABIEs	48
6.5.2 JSON schema fragment for Library ABIEs	48
6.5.3 JSON schema fragment for BBIEs	49
6.5.4 JSON schema declarations for ABIEs	50
6.5.5 JSON schema declarations for ASBIEs	52
6.5.6 JSON schema declarations for BBIEs	53
6.5.7 JSON schema declarations for Qualified Data Types	54
6.6 Extension JSON schema fragments and declarations	54
6.6.1 Extension information in JSON	54
6.6.2 Extension collection JSON schema fragments and declarations	55
6.6.3 JSON schema fragment for the extension content data type declaration	58
6.7 Qualified data types JSON schema fragment and declarations	58
6.8 Unqualified data types JSON schema fragment and declarations	59
6.9 CCTS Core Component Types JSON schema	62
7 Conformance	64

Appendixes

A (informative) Release notes	68
A.1 Availability	68
A.2 Status of this release	68
A.3 Package structure	68
A.4 Release history	68
B (informative) Acknowledgements	70
C (informative) Additional production processes	71
C.1 CCTS serialization	71
C.2 Reporting	71
D Temporary Annex - Change Log (Non-Normative)	72

1 Introduction

1.1 Basic concepts

1.1.1 Naming and design rules

The Open-edi Reference Model [\[ISO 14662\]](#) defines an information bundle as the abstract description of the structure and semantics of the information exchanged between parties.

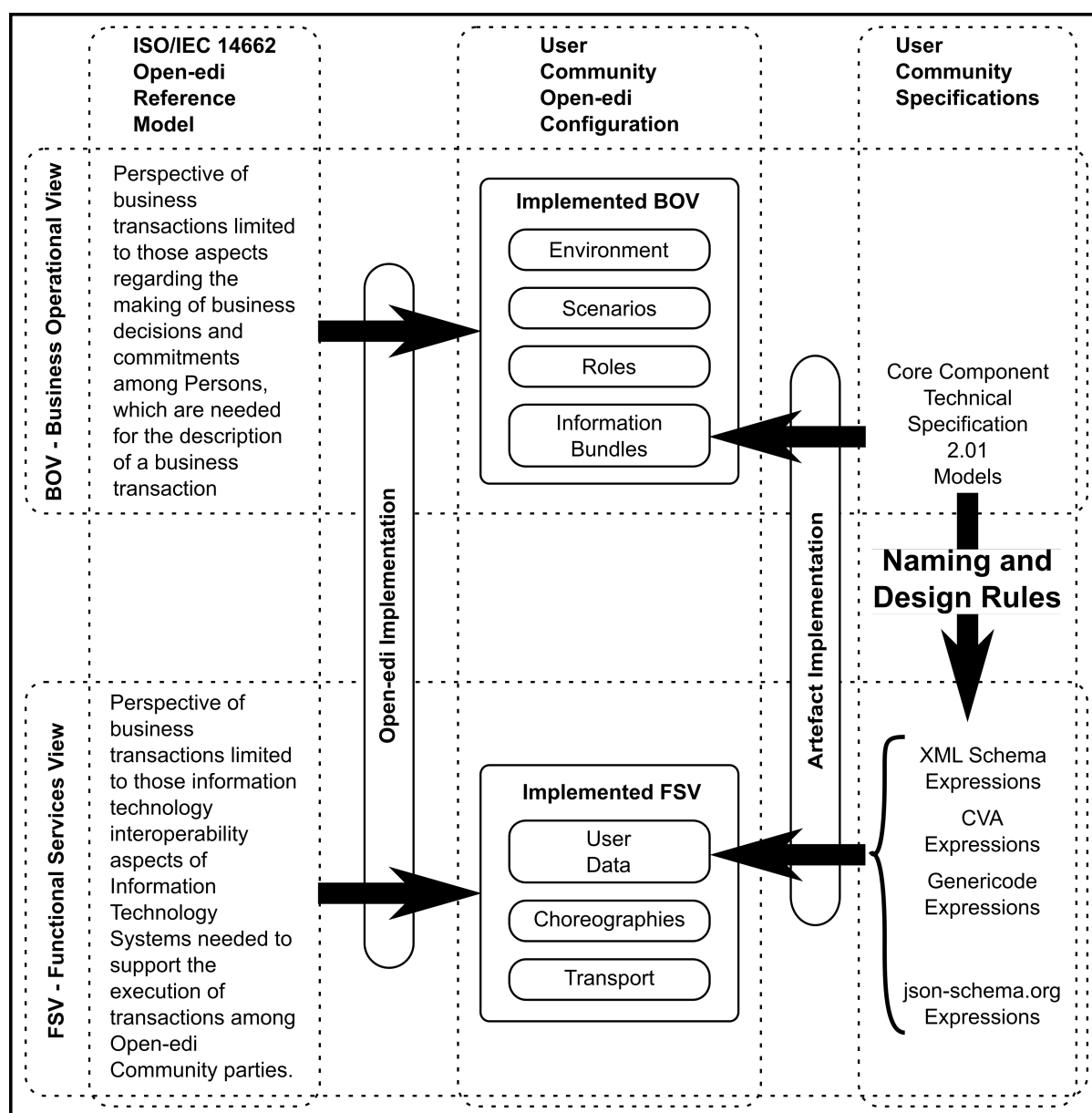
An information bundle can be represented as a logical semantic model. This logical semantic model can be used to produce a physical syntax model that defines the structure and syntax of the Open-edi user data actually exchanged in business transactions. These naming and design rules formalize a method of representing the semantic components of information bundles using the ebXML Core Components Technical Specification [\[CCTS\]](#).

These semantic models can then be used to produce equivalent W3C Schema XSD files [\[XSD schema\]](#) and OASIS Context/value Association [\[CVA\]](#) expressions suitable for defining and validating XML documents [\[XML\]](#) used to convey Open-edi user data.

Also, or alternatively, these semantic models can then be used to produce equivalent JSON schema [\[JSON Schema\]](#) expressions suitable for defining and validating JSON documents [\[ISO 21778 - ECMA JSON\]](#) used to convey Open-edi user data.

This is illustrated in [Figure 1, "Naming and Design Rules in an Open-edi Application"](#).

Figure 1. Naming and Design Rules in an Open-edl Application



The rules presume the reader is already familiar with the following specifications:

- United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) Core Components Technical Specification 2.01 – Part 8 of the ebXML Framework [[CCTS](#)];
- ISO/IEC 11179 Data Elements [[ISO 11179](#)];
- W3C Schema [[XSD schema](#)] XSD files for XML document constraint specification;
- OASIS Context/value association using genericcode [[CVA](#)] for code list association;
- OASIS genericcode [[genericcode](#)] for code list representation;
- JSON data interchange format [[ISO 21778 - ECMA JSON](#)]; and
- JSON Schema [[JSON Schema](#)].

Note

The OASIS Universal Business Language (UBL) can be considered as a reference implementation of these naming and design rules and the examples used in this specification are primarily taken from the UBL 2.2 specifications.

Note

Other validation artefacts (using RelaxNG and ASN.1) are also provided for UBL 2.1. The rules for producing these are outside the scope of this work product.

Note

The direction taken regarding the JSON implementation of these naming and design rules was developed in the discussion paper [[CCTS-and-JSON](#)] where the principles are described with instance examples.

1.1.2 Modeling concepts

Information bundles (describing information to be exchanged) are modeled as a collection of semantic components.

Each semantic component in an information bundle corresponds to one of the following types of Business Information Entity (BIE) in a CCTS Document Model:

- a single irreducible semantic component (referred to as a Basic Business Information Entity or BBIE);
- a structured aggregation of other semantic components (referred to as an Aggregate Business Information Entity or ABIE); or
- an association between ABIEs (referred to as an Association Business Information Entity or ASBIE).

The CCTS semantic model is not dependent on the syntax of the Open-edi user data actually exchanged.

The Open-edi user data exchanged between parties is a machine-readable instance of an information bundle (CCTS Document Model).

This specification assumes XML is the machine-readable syntax used for exchanging the Open-edi user data.

When using an XML document for exchanging Open-edi user data each BIE corresponds to an XML element.

The relationships between these various concepts are described in [Table 1, "Modeling concepts"](#)

Table 1. Modeling concepts

Open-edi Reference Model (ISO 14882)	Semantic Model (ebXML CCTS)	Physical representations (W3C XML; ECMA JSON)
Information bundle	Semantic model	Document schema
Semantic component	Business Information Entity	XML element; JSON object
Open-edi user data	Document	XML document; JSON document
	Document ABIE	XML document element schema definition with only element children, and the XML document element itself; JSON document object schema definition with only object children, and the top-most object itself
	Library ABIE	XML element schema definitions (but not a document element schema definition) with only element children; JSON object schema definition (but not a top-level object schema definition) with only object array children
	ASBIE	XML element (but not a document element) with only element children, defined by a Library ABIE; JSON object (but not a top-level object) with only object array children, defined by a Library ABIE
	BBIE	XML element with only text characters as children, no elements as children, as defined by a BBIE; JSON object with only object array children, defined by a BBIE

1.1.3 Business Information Entities

Following the conventions of the ebXML Core Component Technical Specification [[CCTS](#)] a Business Information Entity (BIE) is piece of business data or a group of pieces of business data with a unique business semantic definition.

A BIE is the result of using a core component within a specific business context. As such each BIE must be based on a core component. The definition of core components is outside the scope of this specification.

It is the Business Information Entities that provide the structure for the semantic components of the body of the business document.

The semantic components used to create the Open-edi user data validation artefacts (following these naming and design rules) are to be applied in a specific business context. Therefore it is the contextualized Business Information Entities to which these rules apply and not the core components from which they have been derived.

1.2 Terminology

1.2.1 Key words

The key words *must*, *must not*, *required*, *shall*, *shall not*, *should*, *should not*, *recommended*, *may*, and *optional* in this document are to be interpreted as described in [[RFC 2119](#)]. Note that for reasons of style, these words are not capitalized in this document.

1.2.2 Terms and Definitions

Business Information Entity (BIE)

A piece of business data or a group of pieces of business data with a unique Business Semantic definition, see [\[CCTS\]](#).

Context/value Association (CVA)

The association of value constraints imposed on information found in a particular document context, see [\[CVA\]](#).

document ABIE

The apex ABIE of an information bundle.

document element

The apex element of information in an XML document, see [\[XML\]](#).

extension collection

The set of extension elements found in an XML document, constrained by an extension schema, that supplements the base information that is constrained by the published document model.

extension item

A single instance of structured supplemental information and its associated metadata distinguishing it from other extension items.

extension point

The apex element of structured supplemental information described by its metadata.

information bundle

The formal description of the semantics of the recorded information to be exchanged, as defined in [\[ISO 14662\]](#).

JSON schema

A JSON vocabulary to annotate and validate JSON documents [\[JSON Schema\]](#).

naming and design rules

A set of rules governing the expression of information bundles using [\[CCTS\]](#), and the creation of either or both the associated XML document model validation artefacts using [\[CVA\]](#) and [\[XSD schema\]](#), or/and JSON schema model artefacts using [\[JSON Schema\]](#).

object class

A set of ideas, abstractions, or things in the real world that are identified with explicit boundaries and meaning and whose properties and behavior follow the same rules [\[ISO 11179\]](#) (definition 3.3.22).

Open-edi user data

Machine-readable instance of the content of information bundles or components of information bundles (as semantic components), see [\[ISO 14662\]](#).

property

A characteristic common to all members of an object class [\[ISO 11179\]](#) (definition 3.3.29).

semantic component

A unit of information unambiguously defined in the context of the business goal of the business transaction. A semantic component may be atomic or composed of other semantic components, see [\[ISO 14662\]](#).

XSD schema

An XML document model definition conforming to the W3C XML Schema language [\[XSD1\]](#)[\[XSD2\]](#).

The terms Core Component (CC), Basic Core Component (BCC), Aggregate Core Component (ACC), Association Core Component (ASCC), Business Information Entity (BIE), Basic Business Information Entity (BBIE), and Aggregate Business Information Entity (ABIE) are used in this specification with the meanings given in [CCTS].

The terms Object Class, Property Term, Representation Term, and Qualifier are used in this specification with the meanings given in [ISO 11179].

1.2.3 Symbols and Abbreviations

ABIE

Aggregate Business Information Entity [CCTS]

ASBIE

Association Business Information Entity [CCTS]

BIE

Business Information Entity [CCTS]

BBIE

Basic Business Information Entity [CCTS]

CCTS

Core Component Technical Specification [CCTS]

CVA

Context/value Association [CVA]

JSON

JavaScript Object Notation [ISO 21778 - ECMA JSON]

NDR

Naming and Design Rules [naming and design rules](#)

TC

Technical Committee

XSD

W3C XML Schema Definition [XSD schema](#)

1.3 Normative References

[CCTS] *UN/CEFACT Core Component Technical Specification - Part 8 of the ebXML Framework* 15 November 2003 Version 2.01 http://www.unece.org/fileadmin/DAM/cefact/codesfor-trade/CCTS/CCTS_V2-01_Final.pdf

[CVA] *OASIS Context/value association using genericcode 1.0*. 15 April 2010. Committee Specification 01. <http://docs.oasis-open.org/codelist/cs01-ContextValueAssociation-1.0/doc/context-value-association.html>.

[genericcode] *OASIS Code List Representation (Genericcode) Version 1.0*. 28 December 2007. Committee Specification 01. <http://docs.oasis-open.org/codelist/cs-genericcode-1.0/doc/oasis-code-list-representation-genericcode.html>.

[ISO 11179] *ISO/IEC 11179-1:2004 Information technology — Specification and standardization of data elements — Part 1: Framework for the specification and standardization of data elements* [http://standards.iso.org/ittf/PubliclyAvailableStandards/c035343_ISO_IEC_11179-1_2004\(E\).zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c035343_ISO_IEC_11179-1_2004(E).zip)

- [ISO 14662] ISO/IEC 14662:2004 Information technology — Open-edi reference model <http://standards.iso.org/ittf/PubliclyAvailableStandards/>
- [ISO 21778 - ECMA JSON] ISO/IEC 21778 Information technology — The JSON data interchange format <http://standards.iso.org/ittf/PubliclyAvailableStandards/>, ECMA 404 The JSON data interchange format <https://www.ecma-international.org/publications/standards/Ecma-404.htm>
- [JSON Schema] JSON Schema Validation: A Vocabulary for Structural Validation of JSON, A. Wright, G. Luff, Editors, <http://json-schema.org/latest/json-schema-validation.html>
- [RFC 2119] S. Bradner, , Key words for use in RFCs to Indicate Requirement Levels, BCP 14, RFC 2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [XML] Extensible Markup Language (XML) 1.0 (Fifth Edition), T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau, Editors, W3C Recommendation, 26 November 2008, <http://www.w3.org/TR/2008/REC-xml-20081126/>. Latest version available at <http://www.w3.org/TR/xml>.
- [XPath 1.0] XML Path Language (XPath) Version 1.0, J. Clark, S. DeRose, Editors, W3C Recommendation, 16 November 1999, <http://www.w3.org/TR/1999/REC-xpath-19991116/>. Latest version available at <http://www.w3.org/TR/xpath>.
- [XSD1] XML Schema Part 1: Structures Second Edition, H. S. Thompson, D. Beech, M. Maloney, N. Mendelsohn, Editors, W3C Recommendation, 28 October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>. Latest version available at <http://www.w3.org/TR/xmlschema-1>.
- [XSD2] XML Schema Part 2: Datatypes Second Edition, P. V. Biron, A. Malhotra, Editors, W3C Recommendation, 28 October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>. Latest version available at <http://www.w3.org/TR/xmlschema-2>.

1.4 Non-Normative References

- [CCTS-and-JSON] CCTS and JSON discussion paper, 06 December 2016, Kenneth Bengtsson, Erlend Bergheim, G. Ken Holman, Editors, https://www.oasis-open.org/committees/document.php?document_id=59528.
- [UBL-2.1] Universal Business Language Version 2.1. 04 November 2013. OASIS Standard. <http://docs.oasis-open.org/ubl/os-UBL-2.1/UBL-2.1.html>.
- [UBL-CCTS] UBL Conformance to ebXML CCTS ISO/TS 15000-5:2005 Version 1.0 08 May 2014. Committee Note 01. <http://docs.oasis-open.org/ubl/UBL-conformance-to-CCTS/v1.0/>.
- [xmldsig] XML-Signature Syntax and Processing Version 1.1, D. E. Eastlake, J. Reagle, D. Solo, F. Hirsch, M. Nyström, T. Roessler, K. Yiu, Editors, W3C Recommendation, 11 April 2013, <http://www.w3.org/TR/2013/REC-xmldsig-core1-20130411/>. Latest version available at <http://www.w3.org/TR/xmldsig-core1>.

2 Context of use and application of these rules

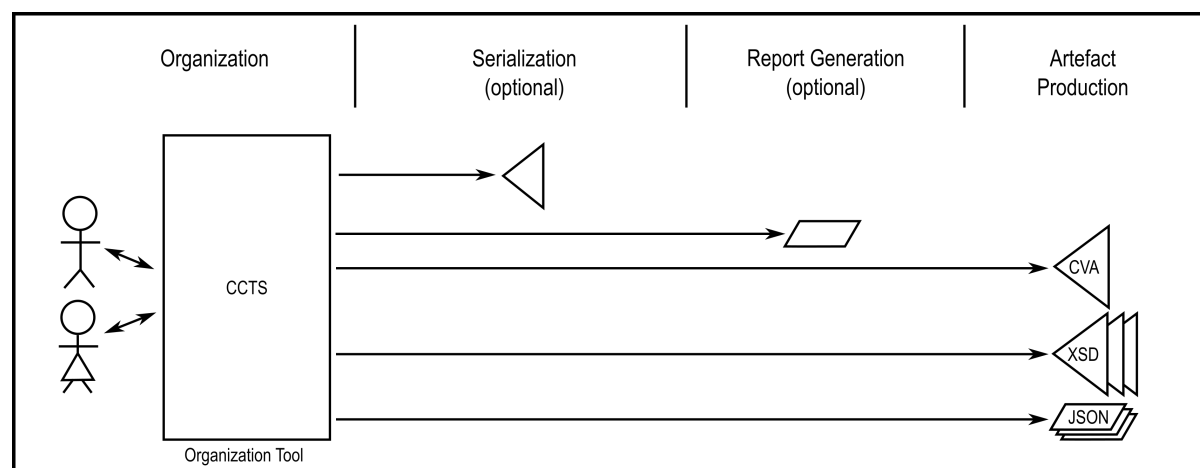
These XML Naming and Design Rules may be used to create a collection of artefacts for defining and validated a set of extensible XML document types and extensible JSON schema definitions.

They describe processes for:

- A. expressing the semantic components of information bundles using the ebXML Core Components Technical Specification [[CCTS](#)], and
- B. producing XML definition and validation artefacts based on those semantic components, specifically:
 - I. XML document structural constraints of elements and attributes (for example, nesting and cardinality) using W3C Schema [[XSD schema](#)],
 - II. XML data value constraints using OASIS Context/value Association [[CVA](#)] expressions of values [[genericcode](#)] (for example, coded value domains or code lists) with XML document locations using XPath [[XPath 1.0](#)], and
 - III. JSON expression constraints using JSON Schema [[JSON Schema](#)] expressions.

These processes are depicted in [Figure 2, “Generic NDR processes to create validation artefacts”](#).

Figure 2. Generic NDR processes to create validation artefacts



Designers (and implementers) may choose to adopt other, optional processes to produce additional artefacts. For example, the serialization of the information bundle (as a CCTS model) into a form suitable for further processing or the production of reports useful in the design and review of the model. See [Appendix C, \(informative\) Additional production processes](#) for more details.

3 Information modeling

3.1 Document ABIEs

A Document ABIE structures the apex of the information bundle to be exchanged between parties.

MOD01 Document ABIE
The apex of the information bundle shall be structured as a single top-level ABIE, referred to in this specification as a Document ABIE.
Note The rationale is that the Document ABIE is always considered a one-member collection in and of itself with no other members in the collection.

3.2 ABIE library

The ABIE library is a collection of common, reusable ABIEs available to be referenced by ASBIEs.

The ABIE library does not include any Document ABIEs.

MOD02 ABIE library contents
A Document ABIE shall not be referenced by any ASBIE.
Note The rationale is that Document ABIEs are identified in syntax implementations separately from other collections of ABIEs.

MOD03 ABIE library ordering
The ABIE library shall have all ABIEs defined in alphabetical order of the ABIE's dictionary entry name.
Note The rationale is that the library can be very large and a reader new to the library may be unfamiliar with any arbitrary ordering of the ABIEs. Designers and implementers can navigate a collection of ABIEs more easily when they are in alphabetical order.

3.3 Extensions

Wherever possible semantic components within an information bundle should be expressed using the BIEs found in an existing Document ABIE or the ABIE Library. However there may be implementations where supplementary information is required to be exchanged in the information bundle in a way that does not interfere with existing BIEs.

The extension mechanism in this specification provides for including additional semantic components that may augment a standardized information bundle with customized additional information.

Extensions can contain new BIEs or can reference BIEs from existing ABIEs but used in a different context. Extensions can also include foreign constructs not defined as BIEs.

Extensions may be horizontal in nature in that they are available to use for all information bundles. An example might be the structure of digital signatures.

Extensions may also be vertical in nature, in that they are applicable only to a single information bundle. For example, additional invoice line detail information to augment an invoice for a specific industry.

An extension collection provides a placeholder for the extensions to be used with a set of Open-edi user data. Within each collection there may be a number of extensions, each with metadata properties regarding the extension and each with a single extension point (the apex structure of the supplemental information).

MOD04 Extension availability
Each document shall allow for optional augmentation with a collection of information not conceptually described by existing BIEs.

Note

There are no constraints on the information that may be included in an extension, including re-using BIEs from existing ABIEs.

Note

When using an XML document for exchanging Open-edi user data, extension XSD schema fragments augment the document's XSD schema created for a document ABIE.

4 Dictionary information

4.1 Dictionary information overview

A BIE is described by the values of its dictionary information as specified by the Core Components Technical Specification 2.01 [CCTS].

To facilitate the definition of the appropriate name for a BBIE, the CCTS property term is specified as the concatenation of two contributing dictionary information values. The optional property term possessive noun and the mandatory property term primary noun are used. When the possessive noun is used the values are separated by a space character.

To facilitate the definition of the appropriate name for a ASBIE, the CCTS property term is specified as the concatenation of two contributing dictionary information values. The optional associated object class qualifier and the mandatory associated object class are used. When the associated object class qualifier is used the qualifier value is suffixed with an underscore character and the values are separated by a space character.

4.2 Dictionary information values

Certain rules govern the creation and expression of dictionary information values for all BIEs defined in the semantic model of an information bundle.

COM01 Dictionary information values
The text describing dictionary information values shall be a string value of Unicode characters without embedded hierarchical structure. The value itself may be structured in its syntax within the string.
Note
The rationale is that the dictionary information values may be constrained in its expression, such as is true in an XML attribute.

COM02 Dictionary information value prohibited characters and character sequences
The following characters shall not be contained in any dictionary information value:
<ul style="list-style-type: none">the characters “<”, “>”, “&”white-space characters other than the “ ” (space) characterthe character sequence “--”
Note
The rationale is that prohibiting these characters and sequences will allow the dictionary information values to be processed more simply in different XML contexts without special handling.
Note
The constraint on white-space characters prohibits values from being structured as paragraphs.

4.3 Abbreviations

It may be convenient to abbreviate complex terms or to use commonly-accepted abbreviations in dictionary information values.

COM03 Controlled list of abbreviations in BIE names
Abbreviations for terms used in BIE names shall be taken from a controlled list of abbreviations agreed for use within the semantic model.
Examples
“Identifier” is abbreviated as “ID”. “Universally Unique Identifier” is abbreviated as “UUID”.
Note
The rationale is that some common or complex terms have commonly-accepted abbreviations suitable for shortening the length of BIE names.

COM04 Controlled list of abbreviations in dictionary entry name information values
Abbreviations for terms used in dictionary entry name information values shall be taken from a controlled list of commonly-agreed abbreviations.
Examples
“XML Path Language” is abbreviated as “XPath”. “Card Verification Value” is abbreviated as “CV2”.
Note
The rationale is that some common terms have widely-accepted abbreviations in general use or in particular use within the information domain. Inconsistent use of abbreviations may lead to semantic ambiguity.

COM05 List of equivalent terms in BIE names
Equivalent terms used in BIE names shall be taken from a list of commonly-agreed equivalent terms.
Example
The property term primary noun “URI” is considered equivalent to the representation term “Identifier”.
Note
The rationale is that some common terms wholly include the concepts presented in other terms and so should be considered equivalent in order to prevent duplication.

4.4 Dictionary information for BIEs

4.4.1 Component type for BIEs

COM06 Component Type for a BIE

Each BIE component shall be typed as being one of either an ABIE, BBIE or ASBIE.

4.4.2 Dictionary information for ABIEs

COM07 Minimum set of dictionary information values describing an ABIE

Each ABIE shall have the following set of dictionary information values:

- | |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none">• Component Type (mandatory)<ul style="list-style-type: none">• shall be the value "ABIE"• Definition (mandatory)<ul style="list-style-type: none">• this value shall describe the ABIE using complete natural language sentences in a single paragraph• Alternative Business Terms (optional)<ul style="list-style-type: none">• this value shall list any other commonly used terms that are synonyms for the ABIE• Object Class Qualifier (optional)<ul style="list-style-type: none">• this value shall qualify the object class for a specific context• Object Class (mandatory)<ul style="list-style-type: none">• this value shall identify the object of interest within an information bundle; it is the class to which the ABIE's BIEs belong• Name (mandatory)<ul style="list-style-type: none">• this value shall be the concatenation of Object Class Qualifier and the Object Class without any spaces, abbreviating the values as required |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Example (using an XPath expression)

Given the following:

\$OCQ = Object Class Qualifier

\$OC = Object Class

C:ABBREV(arg) = custom function to return the abbreviation of an argument, or the argument itself if no abbreviation, and all spaces removed

```
concat( C:ABBREV( $OCQ ) ,  
        C:ABBREV( $OC )  
      )
```

- | |
|-------------------------------------------------------------------------------------|
| <ul style="list-style-type: none">• Dictionary Entry Name (mandatory) |
|-------------------------------------------------------------------------------------|

COM07 Minimum set of dictionary information values describing an ABIE

- this value shall be the concatenation of the Object Class Qualifier, followed by an underscore and space when defined, followed by the Object Class, followed by a period and space, followed by the word “Details”

Example (using an XPath expression)

Given the following:

\$OCQ = Object Class Qualifier

\$OC = Object Class

```
concat( if( $OCQ )
        then concat($OCQ,'_ ') else '',
        $OC,
        '. Details'
      )
```

Example ABIE

Fixed value:

Component Type=“ABIE”

Entered values:

Definition=“A class to define common information related to an address.”

Object Class=“Address”

Derived values:

Name=“Address”

Dictionary Entry Name=“Address. Details”

4.4.3 Dictionary information for BBIEs

COM08 Minimum set of dictionary information values describing a BBIE

Each BBIE shall have the following set of dictionary information values:

- Component Type (mandatory)
 - shall be the value “BBIE”
- Cardinality (mandatory)
 - shall be one of:
 - “1” (required and not repeatable),
 - “0..1” (optional and not repeatable),
 - “0..n” (optional and repeatable) or
 - “1..n” (required and repeatable)
- Definition (mandatory)

COM08 Minimum set of dictionary information values describing a BBIE

- this value shall describe the BBIE using complete natural language sentences in a single paragraph
- Alternative Business Terms (optional)
 - this value shall list any other commonly used terms that are synonyms for the ABIE
- Object Class Qualifier (optional)
 - this value shall qualify the object class for a specific context
- Object Class (mandatory)
 - this value shall identify the object class of the ABIE to which the BBIE belongs
- Property Term Qualifier (optional)
 - this value shall qualify the property term for a specific context
- Property Term Possessive Noun (optional)
 - this value shall identify a distinguishing nature of the characteristic of the object class
- Property Term Primary Noun (mandatory)
 - this value shall identify the principle nature of the characteristic of the object class
- Property Term (mandatory)
 - this value shall identify a characteristic of the object class as the concatenation of Property Term Possessive Noun followed by a space should it exist, followed by the Property Term Primary Noun

Example (using an XPath expression)

Given the following:

\$PTPSN = Property Term Possessive Noun

\$PTPRN = Property Term Primary Noun

```
concat( $PTPSN,  
        if( $PTPSN )  
          then ' ' else '',  
        $PTPRN  
      )
```

- Representation Term (mandatory)
 - this value shall identify the form of the value domain and shall be selected from the set of primary and secondary representation terms specified in CCTS Table 8.3 Permissible Representation Terms (ordered by primary term with secondary terms in parentheses):
 - Amount

COM08 Minimum set of dictionary information values describing a BBIE

- Binary Object (Graphic, Picture, Sound, Video)
- Code
- Date Time (Date, Time)
- Identifier
- Indicator
- Measure
- Numeric (Value, Rate, Percent)
- Quantity
- Text (Name)
- Name (mandatory)
 - this value shall be the concatenation of Property Term Qualifier, Property Term Possessive Noun and, when the Property Term Primary Noun is not "Text" or it is "Text" and both the Property Term Qualifier and the Property Term Possessive Noun are not defined, then the Property Term Primary Noun (abbreviating it as required) and, when the Representation Term is not "Text" and the Property Term Primary Noun is not equivalent to the Representation Term, then also the Representation Term component (abbreviating it as required), all without intervening spaces

Example (using an XPath expression)

Given the following:

\$PTQ = Property Term Qualifier

\$PTPSN = Property Term Possessive Noun

\$PTPRN = Property Term Primary Noun

\$RT = Representation Term

C:ABBREV(arg) = custom function to return the abbreviation of an argument, or the argument itself if no abbreviation, and all spaces removed, or the argument itself if no abbreviation, and all spaces removed

C:EQUIVALENT(noun,term) = custom function to return TRUE/FALSE if the primary noun and representation term are to be considered equivalent

```
concat( C:ABBREV($PTQ),  
        C:ABBREV($PTPSN),  
        if( $PTPRN!='Text' OR  
            ( not($PTQ) AND not($PTPSN) ) )  
        then C:ABBREV($PTPRN) else '',  
        if( $RT!='Text' AND not(C:EQUIVALENT($PTPRN,$RT)) )  
        then C:ABBREV($RT) else ''  
    )
```

- Dictionary Entry Name (mandatory)
 - this value shall be the concatenation of the Object Class Qualifier, followed by an underscore and space when defined, followed by the Object Class, followed by a period and space, followed by the Property Term Qualifier, followed by an

COM08 Minimum set of dictionary information values describing a BBIE

underscore and space when defined, followed by the Property Term, and then, when either the Property Term Qualifier is defined or the Property Term is not equal to the Representation Term, followed by a period and space and the Representation Term

Example (using an XPath expression)

Given the following:

\$OCQ = Object Class Qualifier
\$OC = Object Class
\$PTQ = Property Term Qualifier
\$PT = Property Term
\$RT = Representation Term

```
concat( $OCQ,  
        if( $OCQ )  
          then '_' else '',  
        $OC,  
        '. ',  
        $PTQ,  
        if( $PTQ )  
          then '_' else '',  
        $PT,  
        if( $PTQ OR ( $PT != $RT ) )  
          then concat( '. ', $RT ) else ''  
      )
```

- Data Type Qualifier (optional)
 - this value shall distinguish particular restrictions on a data type from the use of a data type with other (or no) restrictions
- Data Type (mandatory)
 - this value shall be the concatenation of the Data Type Qualifier followed by an underscore and space when it exists, the Representation Term, followed by a period and space, followed by the word "Type"

Example (using an XPath expression)

Given the following:

\$DTQ = Data Type Qualifier
\$RT = Representation Term

```
concat( $DTQ,  
        if( $DTQ )  
          then '_' else '',  
        $RT,  
        '. Type'  
      )
```

COM08 Minimum set of dictionary information values describing a BBIE**Example BBIE**

Fixed value:

Component Type="BBIE"

Entered values:

Cardinality="0..1"

Definition="An additional street name used to further clarify the address."

Object Class="Address"

Property Term Qualifier="Additional"

Property Term Possessive Noun="Street"

Property Term Primary Noun="Name"

Representation Term="Name"

Derived values:

Name="AdditionalStreetName"

Dictionary Entry Name="Address. Additional_ Street Name. Name"

Property Term="Street Name"

Data Type="Name. Type"

4.4.4 Dictionary information for ASBIEs

COM09 Minimum set of dictionary information values describing an ASBIE

Each ASBIE shall have the following set of dictionary information values:

- Component Type (mandatory)
 - shall be the value "ASBIE"
- Cardinality (mandatory)
 - shall be one of:
 - "1" (required and not repeatable),
 - "0..1" (optional and not repeatable),
 - "0..n" (optional and repeatable) or
 - "1..n" (required and repeatable)
- Definition (mandatory)
 - this value shall describe the BBIE using complete natural language sentences in a single paragraph
- Alternative Business Terms (optional)
 - this value shall list any other commonly used terms that are synonyms for the ABIE
- Object Class Qualifier (optional)
 - this value shall qualify the object class for a specific context

COM09 Minimum set of dictionary information values describing an ASBIE

- Object Class (mandatory)
 - this value shall identify the object class of the ABIE to which the BBIE belongs
- Associated Object Class Qualifier (optional)
 - this value shall qualify the object class of the associated ABIE for a specific context
- Associated Object Class (mandatory)
 - this value shall identify the object class of the ABIE the ASBIE associates to
 - the ABIE must exist in the model with the same qualification (or lack thereof) as the ASBIE's associated object class qualifier
- Property Term Qualifier (optional)
 - this value shall qualify the property term for a specific context
- Property Term (mandatory)
 - this value shall be the concatenation of the Associated Object Class Qualifier, an underscore and a space when defined, and the Associated Object Class

Example (using an XPath expression)

Given the following:

\$AOCQ = Associated Object Class Qualifier
\$AOC = Associated Object Class

```
concat( $AOCQ,  
        if( $AOCQ )  
          then '_ ' else '',  
        $AOC  
      )
```

- Representation Term (mandatory)
 - this value shall be the same as the Property Term
- Name (mandatory)
 - this value shall be the concatenation of Property Term Qualifier and the Property Term without any spaces or underscore, abbreviating the values as required

Example (using an XPath expression)

Given the following:

\$PTQ = Property Term Qualifier
\$PT = Property Term
C:ABBREV(arg) = custom function to return the abbreviation of an argument, or the argument itself if no abbreviation, and all spaces removed, or the argument itself if no abbreviation, and all spaces removed

COM09 Minimum set of dictionary information values describing an ASBIE

```
concat( C:ABBREV($PTQ),  
        C:ABBREV($PT)  
      )
```

- Dictionary Entry Name (mandatory)
 - this value shall be the concatenation of the Object Class Qualifier, followed by an underscore and space when defined, followed by the Object Class, followed by a period and space, followed by the Property Term Qualifier, followed by an underscore and space when defined, followed by the Property Term, and then when the Property Term Qualifier is defined, followed by a period and space and the Representation Term

Example (using an XPath expression)

Given the following:

\$OCQ = Object Class Qualifier
\$OC = Object Class
\$PTQ = Property Term Qualifier
\$PT = Property Term
\$RT = Representation Term

```
concat( $OCQ,  
        if( $OCQ )  
          then '_ ' else '',  
        $OC,  
        '. ',  
        $PTQ,  
        if( $PTQ )  
          then '_ ' else '',  
        $PT,  
        if( $PTQ )  
          then concat('. ', $RT) else ''  
      )
```

Example ASBIE

Fixed value:

Component Type="ASBIE"

Entered values:

Cardinality="0..1"

Definition="The buyer of the item."

Object Class="Forecast"

Associated Object Class="Customer Party"

Property Term Qualifier="Buyer"

Derived values:

Name="BuyerCustomerParty"

Dictionary Entry Name="Forecast. Buyer_ Customer Party. Customer Party"

Property Term="CustomerParty"

COM09 Minimum set of dictionary information values describing an ASBIE
Representation Term="CustomerParty"

4.4.5 Dictionary entry names

COM10 Dictionary entry name uniqueness
All dictionary entry names in a semantic model shall be unique.
<p>Note</p> <p>The rationale is that a BIE describes a unique semantic component of business data within a specific business context.</p>

COM11 CCTS dictionary information item name value prohibited characters
All information items contributing to a component's dictionary entry name shall be void of all sensitive dictionary entry name markup characters.
The following characters must not be used in any dictionary information values that contribute to the dictionary entry name:
<ul style="list-style-type: none"> • the characters "." (period) and "_" (underscore) • leading, trailing or multiple sequential " " (space) characters
<p>Note</p> <p>The rationale is that prohibiting these characters in name values prevents ambiguity when assembled into the dictionary entry name using these characters to provide structure.</p>

COM12 Use of leading upper case letter in dictionary entry name values
All words that are not abbreviated in name values contributing to the dictionary entry name shall have a leading upper-case letter and all other letters in lower-case.

4.5 Structure of an ABIE

COM13 ABIE contents cannot be empty
Every ABIE shall contain at least one BIE.

COM14 ABIE children ordering
Every ABIE shall have all of its BBIE children listed before its ASBIE children.

5 XML validation artefacts

5.1 XML validation artefacts overview

These NDR provide for expressing the semantic model of an information bundle as XML artefacts that provide for definition and validation of the structure and content of XML Documents.

There are two types of validation artefacts required for XML Documents:

1. W3C Schemas are used to define and validate *the structure* of elements and data content, and
2. OASIS CVA expressions and OASIS genericcode files are used to define and validate *the values* of data content.

These rules do not require these artefacts to be located in any particular directory structure.

5.2 XML Namespaces

An important aspect of identifying and distinguishing the names used for information in XML documents is by using namespaces. Like-named constructs are distinguished by having different namespaces.

Note

Namespace abbreviations (also used here as namespace prefixes) in these examples are not mandatory and have been selected solely for convenience and consistency. Implementations of these NDR and the documents governed by these NDR are welcome to use any namespace abbreviation or namespace prefix for any of the namespaces defined or referenced.

NAM01 Namespaces for information found in information bundles
<p>BIEs expressed in XML documents shall use the following set of namespaces:</p> <ul style="list-style-type: none">• one namespace for each Document ABIE <p>Note</p> <p>These namespaces are not abbreviated in these examples as they are not imported or included.</p> <ul style="list-style-type: none">• one namespace for all BBIE components <p>Note</p> <p>This namespace is abbreviated in these examples as “cbc” for “common basic components”.</p> <ul style="list-style-type: none">• one namespace for all Library ABIE components <p>Note</p> <p>This namespace is abbreviated in these examples as “cac” for “common aggregate components”.</p> <ul style="list-style-type: none">• one namespace for the extension collection and extension metadata elements

NAM01 Namespaces for information found in information bundles

Note

This namespace is abbreviated in these examples as “ext”.

Each extension has a number of namespaces distinct from the document, library and other extensions.

NAM02 Namespaces for an extension

BIEs expressed in extensions shall use the following set of namespaces:

- | |
|----------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none">• one namespace for the apex ABIE of the extension |
|----------------------------------------------------------------------------------------------------|

Note

This namespace is abbreviated in these examples as “myext1”.

- | |
|--------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none">• one namespace for all BBIE extension components that are not existing library components |
|--------------------------------------------------------------------------------------------------------------------------------------------|

Note

This namespace is abbreviated in these examples as “mycbc1”.

- | |
|-------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none">• one namespace for all ABIE extension components that are not existing Library ABIE components |
|-------------------------------------------------------------------------------------------------------------------------------------------------|

Note

This namespace is abbreviated in these examples as “mycac1”.

Note

The structure of an extension parallels that of a Document ABIE with the distinct apex namespace, BBIE namespace and Library ABIE namespace. Where possible the extension should reuse existing library components that have their own namespaces. This parallel approach makes it easier to consider incorporating extension elements in future versions of the library simply by changing the namespace.

Note

In these abbreviations “my” is used as in the first-person possessive pronoun, and “1” implies one of multiple extension namespaces

In addition to the namespaces for the elements in XML documents, the dictionary information regarding BBIE data types is also distinguished using namespaces.

NAM03 Namespaces for BBIE data types

The expression of data type information supporting BBIE definitions shall use the following set of namespaces:

- | |
|----------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none">• one namespace for all qualified data types |
|----------------------------------------------------------------------------------------------|

NAM03 Namespaces for BBIE data types

Note

This namespace is abbreviated in these examples as “qdt”.

- one namespace for all unqualified data types

Note

This namespace is abbreviated in these examples as “udt”.

- one namespace for CCTS Core Component Type definitions

Note

This namespace is abbreviated in these examples as “ccts-cct”.

Note

These namespaces are used only for dictionary information definition and not for BIEs themselves. As such they never appear in the XML document and are therefore never declared in an XML artefact governed by these NDR.

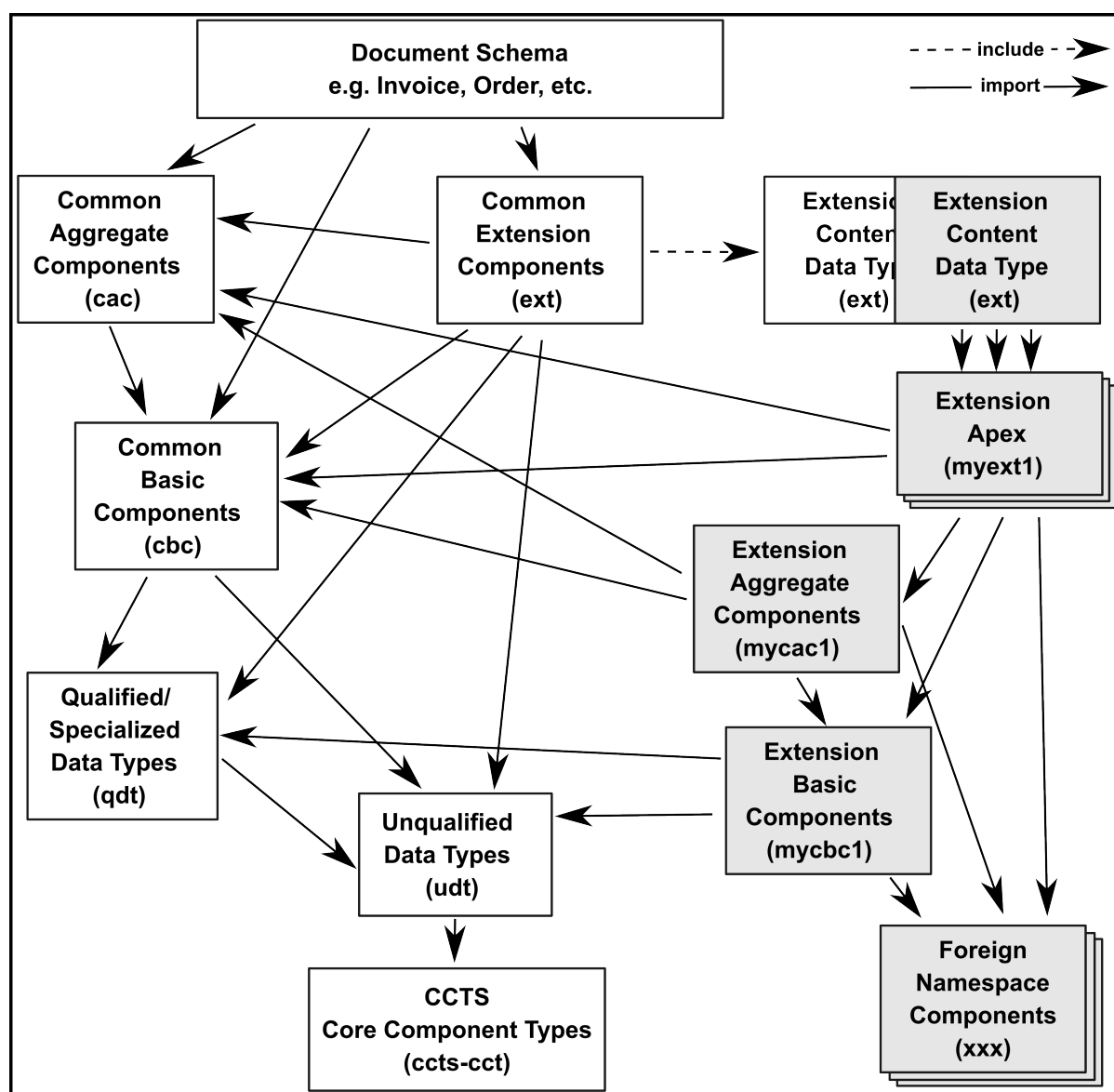
An implementation of these NDR may choose to have additional namespaces for the expression of type information.

5.3 XSD import/include tree

The relationships between the XSD schema fragments that are described in this section are shown in [Figure 3, “Possible import/include hierarchy of XSD schema expressions”](#).

For reference purposes each box is a schema fragment and the parenthesized name tokens identify the namespace associated with the schema fragment.

Figure 3. Possible import/include hierarchy of XSD schema expressions



In this diagram the unshaded boxes represent fragments of the common schema. Once created there is no need for implementers of the schema to modify these fragments. To ensure they are not inadvertently modified, these may be marked as read-only files in the file system.

The shaded boxes represent fragments that extend the common schema. The Extension Content Data Type defines which schemas are in play for the extension point of an extension item in the extension collection. This fragment is distinguished from other fragments in that it is initially created by the designers of the schema and subsequently may be replaced by implementers.

5.4 Schema fragment annotations

There are no constraints regarding what annotation information may be added to the W3C Schema expressions in XSD files.

Good practice suggests augmenting the schema fragments with dictionary information and governance information using W3C Schema declaration annotations and XML comments. This information may be useful to the human reader or to tools exploiting the schema information in providing functionality to

operators, but it does not impact on the interpretation of constraints imposed on XML documents being validated with the XSD documents.

W3C Schema annotations are typically defined for and with each of the many declarations in the XSD file. Good practice suggests providing a version of the published XSD files without annotations so as not to burden runtime processing. A runtime schema processor has no use of informational annotations and may incur unnecessary processing time ingesting and accommodating the information.

Separate from the concept of W3C Schema annotations are simple XML comments that annotate schema. Such comments are ignored by schema processors and do not burden their processing. The information in these comments may be useful to implementers and, in some cases, may be required for intellectual property reasons imposed by the designers. Good practice suggests that such information includes module identification, module revision metadata and copyright declarations. The information in these comments may be useful to implementers and may be required by licensing conditions on the use of the files.

The W3C Schema version annotation (the “version” attribute of the `xsd:schema` element) may be used to record the release version of the collection of schema fragments.

5.5 XML schema fragments and declarations

5.5.1 XML schema fragments for Document ABIEs

FRG01 Document ABIE schema fragments
There shall be one schema fragment created for each Document ABIE.
FRG02 Document ABIE element declaration
Each Document ABIE schema fragment shall include a single element declaration, that being for the Document ABIE.
FRG03 Document ABIE type declaration
Each Document ABIE schema fragment shall include a single type declaration, that being for the content of the Document ABIE.

5.5.2 XML schema fragment for Library ABIEs

FRG04 Library ABIE schema fragment
There shall be one common schema fragment created to contain all ASBIEs (that is, from every Document ABIE and every Library ABIE) and all Library ABIEs. Example In UBL 2.2 the <code>UBL-CommonAggregateComponents-2.2.xsd</code> fragment serves this purpose.
FRG05 Library ABIE element declarations
The common Library ABIE schema fragment shall include an element declaration for every ASBIE in the model (that is, from every Document ABIE and every Library ABIE) and for every Library ABIE.

FRG06 Library ABIE type declarations
The common Library ABIE schema fragment shall include a type declaration for every Library ABIE, each being for the content of each Library ABIE.

There are no constraints on the order of the ABIE declarations in the schema expression.

Note

This lack of an order constraint may seem in conflict with the semantic model library constraint of alphabetical order of ABIE components. Whereas the semantic ABIE components are organized for the benefit of the human reader, the order of the schema component declarations does not affect the schema processor. However, using alphabetical order in the schema fragment may be a convenience for the human reader of the schema expressions.

5.5.3 XML schema fragment for BBIEs

FRG07 BBIE schema fragment
There shall be one common schema fragment created to describe all BBIEs in the model (that is, from every Document ABIE and every Library ABIE).
Example
In UBL 2.2 the <code>UBL-CommonBasicComponents-2.2.xsd</code> fragment serves this purpose.

FRG08 BBIE element declarations
The common BBIE schema fragment shall include an element declaration for every BBIE in the model (that is, from every Document ABIE and every Library ABIE) describing the content of each BBIE.

FRG09 Library ABIE type declarations
The one BBIE schema fragment shall include a type declaration for every BBIE in the model (that is, from every Document ABIE and every Library ABIE), each being for the content of each BBIE.

There are no constraints on the order of the BBIE declarations in the schema expression.

Note

The order of the schema component declarations does not affect the schema processor. However, using alphabetical order in the schema fragment may be a convenience for the human reader of the schema expressions.

5.5.4 XML schema declarations for all BIEs

DCL01 Element declarations
Every BIE element declaration shall be global.
DCL02 Element declaration references
Every BIE element in an ABIE type definition shall be declared by reference.

DCL03 Type declarations

Every BIE type declaration shall be global.

5.5.5 XML schema declarations for ABIEs

DCL04 ABIE element declaration

Every ABIE element shall be declared with the ABIE name as the element name and the ABIE name suffixed with "Type" as the type.

Example

```
<xsd:element name="ApplicationResponse"
             type="ApplicationResponseType" />
```

DCL05 ABIE type declaration

Every ABIE complex type name shall be declared with the name of the ABIE suffixed with "Type" as the name.

Example

```
<xsd:complexType name="ApplicationResponseType">
  (... contents ...)
</xsd:complexType>
```

DCL06 Library ABIE type declaration content order

The members of a Library ABIE shall be ordered as the sequence (in the order the BIEs appear in the semantic model of the ABIE) of all BBIE element references first, followed by all ASBIE references.

Example

```
<xsd:complexType name="CatalogueRequestLineType">
  <xsd:sequence>
    <xsd:element ref="cbc:ID" minOccurs="1" maxOccurs="1"/>
    <xsd:element ref="cbc:ContractSubdivision"
                 minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="cbc:Note" minOccurs="0"
                 maxOccurs="unbounded"/>
    <xsd:element ref="cac:LineValidityPeriod"
                 minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="cac:RequiredItemLocationQuantity"
                 minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="cac:Item" minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
```

DCL07 Document ABIE type declaration content order

The members of a Document ABIE shall be ordered first with a reference to the extension collection element, followed by the sequence (in the order the BIEs appear in the semantic model of the ABIE) of all BBIE element references first, followed by all ASBIE references.

Example

```
<xsd:complexType name="ApplicationResponseType">
  <xsd:sequence>
    <xsd:element ref="ext:UBLExtensions"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="cbc:UBLVersionID"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="cbc:CustomizationID"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="cbc:ProfileID"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="cbc:ProfileExecutionID"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="cbc:ID"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element ref="cbc:UUID"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="cbc:IssueDate"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element ref="cbc:IssueTime"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="cbc:ResponseDate"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="cbc:ResponseTime"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="cbc:Note"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="cbc:VersionID"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="cac:Signature"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="cac:SenderParty"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element ref="cac:ReceiverParty"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element ref="cac:DocumentResponse"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

Note

The rationale for positioning extension information at the very start of the XML instance is to allow processing applications acting sequentially on the document to consume and cache all non-modeled extension information in preparation for consuming any of the modeled document information. Should extension information follow modeled information, the sequential processing of the modeled information would have passed before recognizing the need to associate extension information. In essence, such a sequential processing application would have to cache the entire document, thus losing the benefit of sequential processing.

DCL08 Document ABIE extension element cardinality

In the content type for every Document ABIE the extension collection element cardinality shall be declared as optional and not repeatable.

Example

```
<xsd:element ref="ext:UBLExtensions"
              minOccurs="0" maxOccurs="1"/>
```

5.5.6 XML schema declarations for ASBIEs

DCL09 ASBIE schema element declaration

Every ASBIE element shall be declared with the ASBIE name as the element name and the ABIE name of the associated object class suffixed with "Type" as the type.

Example

```
<xsd:element name="Party" type="PartyType"/>
```

Example

```
<xsd:element name="AgentParty" type="PartyType"/>
```

5.5.7 XML schema declarations for BBIEs

DCL10 BBIE element declaration

Every BBIE element shall be declared with the BBIE name as the element name and the concatenation of the BBIE name and "Type" as the type.

Example

```
<xsd:element name="SourceCurrencyCode"
              type="SourceCurrencyCodeType"/>
```

DCL11 BBIE unqualified type declaration

Every BBIE element type with an unqualified data type shall be declared as simple content restricted from a base of the corresponding unqualified data type without the addition of any additional attributes.

Example

```
<xsd:complexType name="SourceCurrencyBaseRateType">
  <xsd:simpleContent>
    <xsd:restriction base="udt:RateType"/>
  </xsd:simpleContent>
</xsd:complexType>
```

DCL12 BBIE qualified type declaration
Every BBIE element type with a qualified data type shall be declared as simple content restricted from a base of the corresponding qualified data type without the addition of any additional attributes.
Example
<pre><xsd:complexType name="SourceCurrencyCodeType"> <xsd:simpleContent> <xsd:restriction base="qdt:CurrencyCodeType" /> </xsd:simpleContent> </xsd:complexType></pre>

5.5.8 XML schema declarations for Qualified Data Types

DCL13 Qualified data type declaration
Every qualified data type shall be declared as simple content restricted from a base of the corresponding unqualified data type without the addition of any additional attributes.
Example
<pre><xsd:complexType name="CurrencyCodeType"> <xsd:simpleContent> <xsd:restriction base="udt:CodeType" /> </xsd:simpleContent> </xsd:complexType></pre>

5.6 Extension XML schema fragments and declarations

5.6.1 Extension information in XML

The content type for a Document ABIE contains a single optional extension collection element in order to provide for the inclusion of data in the XML that is in addition to the data of the information bundle for the document. Such data may include content designed by other organizations (e.g. signature information) as well as augmentations of the semantic model.

An extension collection element contains one or more extension elements. Each extension element has a suite of metadata elements used to describe the extension. The extension content may reuse existing ABIEs or BBIEs and may contain XML content not modeled as BIEs.

The extension collection and metadata are XML elements implemented as schema fragments and constructs independent of the semantic model.

5.6.2 Extension collection schema fragments and declarations

EXT01 Extension collection schema fragment
The extension collection schema fragment shall include the declarations of the extension collection element, the extension element, the extension content element, the extension metadata elements and any required type information for metadata elements that are not BIEs in the Document ABIEs and Library ABIEs.

EXT01 Extension collection schema fragment

Example

In UBL 2.2 the UBL-CommonExtensionComponents-2.2.xsd fragment serves this purpose.

EXT02 Extension content element declaration

The extension collection schema fragment shall include the declaration of the mandatory extension content element, but not the type information for the extension content element.

Note

The rationale for not including the type information for the extension point element is that the type is subject to change, while the extension collection, the extension item and extension item metadata element and type information is not. This separation allows the extension collection and item schema fragment to be deployed as read-only, while the extension point data type schema fragment can be deployed as writable in order to be defined by users.

EXT03 Extension collection element content

The document's extension collection shall have one or more extension elements as its content.

Example

```
<xsd:complexType name="UBLExtensionsType">
  <xsd:sequence>
    <xsd:element ref="UBLExtension"
      minOccurs="1" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

Note

The rationale is that different users of a document may have different extension items added to the content. Also, different extensions may be thematically distinguished (e.g. the digital signature extension is semantically separate from an extension augmenting invoice line content).

EXT04 Extension element content ordering

The extension element shall declare all available metadata elements (if any) in advance of a last mandatory single extension content element being the extension point under which the extension information is added to the document.

Example

```
<xsd:complexType name="UBLExtensionType">
  <xsd:sequence>
    <xsd:element ref="cbc:ID"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="cbc:Name"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="ExtensionAgencyID"
```

EXT04 Extension element content ordering

```
        minOccurs="0" maxOccurs="1"/>
<xsd:element ref="ExtensionAgencyName"
        minOccurs="0" maxOccurs="1"/>
<xsd:element ref="ExtensionVersionID"
        minOccurs="0" maxOccurs="1"/>
<xsd:element ref="ExtensionAgencyURI"
        minOccurs="0" maxOccurs="1"/>
<xsd:element ref="ExtensionURI"
        minOccurs="0" maxOccurs="1"/>
<xsd:element ref="ExtensionReasonCode"
        minOccurs="0" maxOccurs="1"/>
<xsd:element ref="ExtensionReason"
        minOccurs="0" maxOccurs="1"/>
<xsd:element ref="ExtensionContent"
        minOccurs="1" maxOccurs="1"/>
</xsd:sequence>
</xsd:complexType>
```

Note

There are no constraints on the selection, name, definition or cardinality of the extension metadata elements.

5.6.3 XML schema fragment for the extension content data type declaration

EXT05 Extension content data type schema fragment

The extension content element type schema fragment shall include the declaration of the content type for the extension content element and any import statements defining constraints on recognized constructs.

Example

In UBL 2.2 the `UBL-ExtensionContentDataType-2.2.xsd` fragment serves this purpose.

EXT06 Extension content data type declaration

The extension content element type schema fragment shall contain only a single complex type declaration being a sequence of exactly one element in a namespace other than the extension namespace to be processed with lax validation.

Example

```
<xsd:complexType name="ExtensionContentType">
  <xsd:sequence>
    <xsd:any namespace="##other" processContents="lax"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
```

EXT06 Extension content data type declaration

Note

The rationale for the lax validation is to allow for the extension point to contain, without error, any element for which there are no constraints in any schema fragment imported or included in the validation step.

EXT07 Extension content data type imports

The extension content element type schema fragment may contain import directives for the expected data content of an extension.

Example

```
<xsd:import namespace="urn:oasis:names:specification:ubl:schema:xsd:CommonSignatureComponents-2"
  schemaLocation="UBL-CommonSignatureComponents-2.2.xsd"/>
```

Note

The rationale for including import directives is to validate those constructs found in an extension that are expected.

Note

There is no order to the import directives.

5.7 Qualified data types XML schema fragment and declarations

QDT01 Qualified data types schema fragment

The qualified data types schema fragment shall include the declarations of any qualified data types referenced in the schema fragment for BBIEs.

Example

In UBL 2.2 the UBL-QualifiedDataTypes-2.2.xsd fragment serves this purpose.

QDT02 Qualified data type declaration name

Every qualified data type shall be declared using the name of the data type qualifier followed by the unqualified data type name.

QDT03 Qualified data type declaration basis

Every qualified data type shall be based on an unqualified data type, imposing whatever qualifications are required to be expressed using XSD schema semantics.

Note

In UBL 2.2 there are no qualifications expressed using XSD schema semantics.

QDT04 Qualified data type declaration constraint

Every qualified data type declaration shall be such that every possible instance of the declared type is also an instance of the base type.

Note

This constraint prevents additions of anything that is not part of the base type, such as the introduction of any new attributes or sub-elements, or any less-constrained element or attribute values.

5.8 Unqualified data types XML schema fragment and declarations

UDT01 Unqualified data types schema fragment

The unqualified data types schema fragment shall include the declarations of all unqualified data types referenced in the schema fragment for BBIEs.

Example

In UBL 2.2 the `UBL-UnqualifiedDataTypes-2.2.xsd` fragment serves this purpose.

UDT02 Unqualified data types declaration inclusions

An unqualified data type shall be declared for every one of the permitted Primary Representation Terms and Secondary Representation Terms defined as Permissible Representation Terms in the Core Component Technical Specification [CCTS].

UDT03 Unqualified data types declaration exclusions

Unqualified data types shall only be declared for the permitted Primary Representation Terms and Secondary Representation Terms defined as Permissible Representation Terms in the Core Component Technical Specification [CCTS].

UDT04 Unqualified data types declaration base

Every unqualified data type shall be either a restriction on one of the approved Core Component Types defined in the Core Component Technical Specification [CCTS], or an extension of a base XSD data type.

Note

This constraint may be accomplished by either restricting a declaration imported from the Core Component Types schema or by wholly replacing the corresponding Core Component Types schema declaration.

Note

The rational for having UDT declarations is to impose some XSD syntax semantics on top of the more general decimal and string lexical syntax defined in the CCTS specification of Core Component Types. For example, the CCTS Core Component Type for date and time is a simple string without constraint. Such lax structuring of the field value does not serve users in that no particular format is obligated. The UDT can impose, for example, the `xsd:dateTime` lexical syntax on all date and time values, overriding the CCTS definition.

UDT04 Unqualified data types declaration base

Note

This rule does allow an optional supplementary component defined in the CCTS Core Component Type not to be available in the associated unqualified data type. For example, if the UDT implements a built-in XSD data type for the component then there is no use of a format supplementary component and associated attribute and so the format attribute declaration can be omitted and, thereby, be unavailable for use for that data type.

Example 1

In this example the unqualified data type uses the base type without change:

```
<xsd:complexType name="NumericType">
  <xsd:simpleContent>
    <xsd:restriction base="ccts-cct:NumericType"/>
  </xsd:simpleContent>
</xsd:complexType>
```

Example 2

In this example the unqualified data type redeclares the base type's optional attribute as mandatory:

```
<xsd:complexType name="MeasureType">
  <xsd:simpleContent>
    <xsd:restriction base="ccts-cct:MeasureType">
      <xsd:attribute name="unitCode"
        type="xsd:normalizedString"
        use="required"/>
    </xsd:restriction>
  </xsd:simpleContent>
</xsd:complexType>
```

Example 3

In this example the unqualified data type replaces the base type with no attributes and with an XSD built-in type for content:

```
<xsd:complexType name="DateTimeType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:dateTime"/>
  </xsd:simpleContent>
</xsd:complexType>
```

UDT05 Unqualified data types declaration constraint

Every unqualified data type declaration that is not a formal restriction of one of the Core Component Type declarations defined in the Core Component Technical Specification [CCTS] schema fragment shall be such that every possible instance of the declared type is also an instance of one of the Core Component Types as defined in CCTS.

UDT05 Unqualified data types declaration constraint

Note

This constraint prevents additions of anything that is not part of the base Core Component Type, such as the introduction of any new attributes or sub-elements, or any less-constrained element or attribute values.

Example

In this example the unqualified data type for a date is based on the XSD data type for date and all instances of the date data type are also instances of the string-based `DateTimeType` data type in CCTS:

```
<xsd:complexType name="DateTimeType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:date"/>
  </xsd:simpleContent>
</xsd:complexType>
```

5.9 CCTS Core Component Types XML schema

All primitive data types correspond to the 10 CCTS Primary Representation Terms defined in [\[CCTS\]](#) Section 8-3 “Permissible Representation Terms”.

CCT01 CCTS Core Component Type schema

The core component type schema of primitive data types for primary representation terms on which the unqualified data types are based shall be an unmodified copy of the schema fragment published by UN/CEFACT with the following embedded title and metadata:

CCTS Core Component Type Schema Module

Module of Core Component Type

Agency: UN/CEFACT

VersionID: 1.1

Last change: 14 January 2005

Example

In UBL 2.2 the `CCTS_CCT_SchemaModule-2.2.xsd` fragment serves this purpose.

5.10 XML attribute names

ATT01 Leading name part in attribute names

Every attribute's derived name shall be composed with the leading name part entirely in lower-case, even when that name part is an agreed-upon abbreviation.

Example

The data type of the BBIE known as “Binary Object. Uniform Resource. Identifier” is represented with the attribute named “uri”

ATT01 Leading name part in attribute names**Note**

Terms used in attribute names of supplementary components of CCTS Table 8-2 "Approved Core Component Type Content and Supplementary Components" are subject to abbreviation.

ATT02 Non-leading abbreviations in attribute names

When an attribute's derived name is composed with an agreed-upon abbreviation in other than the leading name part, the abbreviation shall be used unchanged.

Examples

The data type of the BBIE known as "Amount Currency. Identifier" is represented with the attribute named "currencyID"

The data type of the BBIE known as "Amount Currency. Code List Version. Identifier" is represented with the attribute named "currencyCodeListVersionID"

Note

Terms used in attribute names of supplementary components of CCTS Table 8-2 "Approved Core Component Type Content and Supplementary Components" are subject to abbreviation.

5.11 Data type qualifications in XML

Data types may be qualified to define additional constraints on the possible values of the BBIEs of that data type. These constraints may be subject to change over time and so should be applied in a manner that allows modification of the data type qualifications without impacting the schema.

Code lists and identifier lists are examples of controlled sets of values (e.g. currency codes, country codes, product identifiers, etc.).

For some communities of users (e.g. those with business-oriented XML documents) the management of controlled vocabularies presents particular challenges for document modeling. While communities may standardize document structures, trading partners within the community have their own constraints that may change on an hourly basis yet must work within the community framework.

Externalizing the list in a genericcode file expresses the enumeration as a resource available to the application for data entry. However, the choice of genericcode file may vary on a per-information item basis due to the item's document context, or perhaps vary again for particular trading partners. Expressing the appropriate mappings in a colloquial fashion inhibits interoperability and the sharing of resources and program code.

A context/value association file specifies the relationship from information items found in different document contexts to one or more external genericcode files for each item. With these relationships a directed authoring environment can precisely direct the editing of individual information items. Different context/value association files can then be used to create instances for different purposes that have different constraints on the enumerations used therein.

DTQ01 Data type qualification CVA file

Data type qualifications that are not expressed as qualified data types using XSD schema semantics may be expressed using the OASIS Context Value Association [CVA] XML vocabulary.

DTQ01 Data type qualification CVA file

Note

The CVA file provides for users to associate value validation constraint expressions and/or coded domain value enumerations with different CVA contexts. A value validation constraint is expressed using XPath. A coded domain value enumeration is expressed using one or the union of more than one OASIS genericcode file.

Note

The CVA expressions typically are not used at runtime. More likely the CVA expressions and their associated genericcode files would be processed or compiled into a runtime validation artefact. These rules do not constrain where this runtime artefact would be kept, but good practice suggests a location separate from the XSD schemas.

Example

In UBL 2.2 the CVA expression is the `cva/UBL-DefaultDTQ-2.2.cva` file, the associated runtime artefact is the `val/UBL-DefaultDTQ-2.2.xsl` XSLT stylesheet and the referenced genericcode files are located in the `cl/` subdirectory of each of the UBL 2.2 distribution, the UBL 2.1 distribution and the UBL 2.0 distribution.

DTQ02 Data type element content qualifications

A CVA context shall be created for every BBIE with a non-empty value in the CCTS dictionary information for the data type qualifier.

Example

Entered dictionary information values:

Name="ChannelCode"
Data Type Qualifier="Channel"
Representation Term="Code"
Data Type="Channel_ Code. Type"

In this example the constraints on the value of the `cbc:ChannelCode` element are described by the union of three constraint expressions identified by "Channel-2.0", "Channel-2.1" and "Channel-2.2":

```
<Context values="Channel-2.0 Channel-2.1 Channel-2.2"
  metadata="cctsV2.01-code"
  address="cbc:ChannelCode" />
```

Example

In this example the constraints on the value of the `cbc:PayableAmount` element child of the `cac:LegalMonetaryTotal` element are described by the single constraint expression identified by "maxValue":

```
<Context values="maxValue"
  address="cac:LegalMonetaryTotal/
  cbc:PayableAmount" />
```

DTQ03 Data type attribute content qualifications

A CVA context shall be created for every CCTS supplementary component to be validated.

Example

In this example the constraints on the value of the `currencyID` attribute are described by the union of two constraint expressions identified by "Currency-2.0", "Currency-2.1" and "Currency-2.2":

```
<Context values="Currency-2.0 Currency-2.1 Currency-2.2"
  metadata="cctsV2.01-amount"
  address="@currencyID" />
```

DTQ04 Value test constraints

A CVA value test constraint shall be written as an XPath expression.

Example

In this example the constraint on the value is that its numeric value be less than 10,000:

```
<ValueTest xml:id="maxValue"
  test=". < 10000" />
```

DTQ05 Value list constraints

A CVA value list constraint shall reference an OASIS genericcode [\[genericcode\]](#) file.

Example

```
<ValueList xml:id="Channel-2.2"
  uri=".. /cl/gc/default/ChannelCode-2.2.gc" />
```

Note

Each genericcode file defines the metadata associated with the enumeration of values therein. Therefore, the union of multiple genericcode files is required when the constraint includes values from different enumerations associated with distinctive metadata.

DTQ06 Value metadata association

The CVA instance metadata sets shall identify the XML attributes used in Open-edi user data that are associated with the supplementary components of the unqualified data types.

Example

In UBL 2.2 the `UBL-DefaultDTQ-2.2.cva` fragment includes such declarations.

6 JSON validation artefacts

6.1 JSON validation artefacts overview

These NDR provide for expressing the semantic model of an information bundle as JSON Schema [[JSON Schema](#)] artefacts that support the definition and validation of the structure and content of JSON documents.

These rules do not require these artefacts to be located in any particular directory structure.

6.2 JSON preservation of XML namespaces

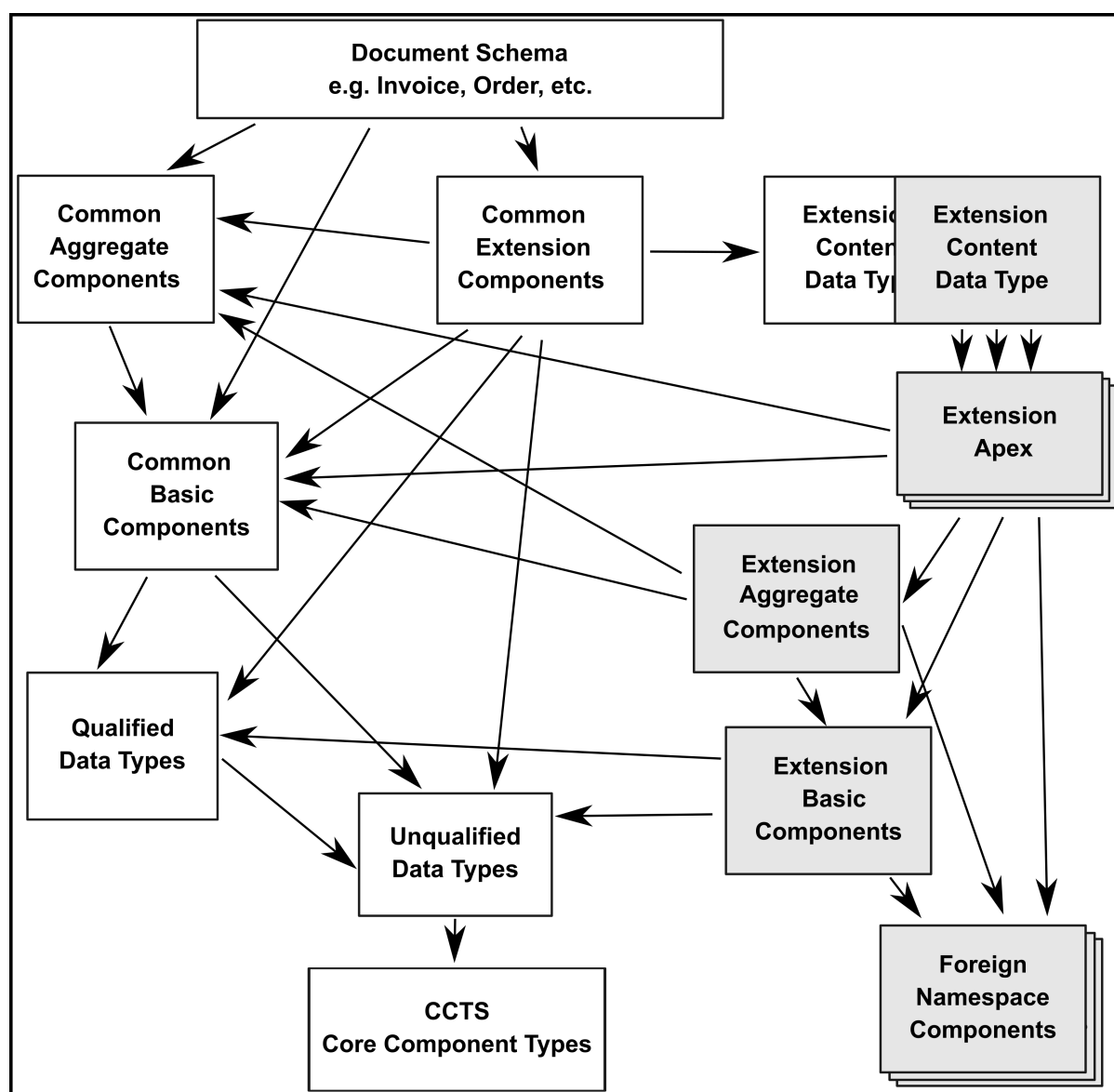
An important aspect of identifying and distinguishing the names used for information in XML documents is by using namespaces. Like-named constructs are distinguished by having different namespaces. These NDR provide for preserving in the JSON structure namespaces from an XML document transliterated to JSON following these rules. Such preservation supports round-tripping, that is, converting the transliterated JSON instance back to an XML document in a lossless fashion, but without consideration for foreign extension content.

NAM21 Namespaces for information found in information bundles
<p>A JSON instance shall provide for preserving the following XML document namespaces for BIEs:</p> <ul style="list-style-type: none">• one namespace for each Document ABIE• one namespace for all BBIE components• one namespace for all Library ABIE components• one namespace for the extension collection and extension metadata elements

6.3 JSON schema file references

The JSON schema expressions are fragmented in order that like-named items in each role can be declared without the use of prefixes or suffixes. This fragmentation mimics that found in the XML schemas illustrated in [Figure 3, "Possible import/include hierarchy of XSD schema expressions"](#).

Figure 4. JSON schema fragment reference tree



6.4 JSON schema fragment annotations

JSON schema expressions provide only for title and description annotations in addition to the value constraint properties. The top-level title and description of each fragment shall include an overall description title for that fragment and any additional global information such as source and copyright details.

Within the fragments, these annotations shall be present only on those declarations of Document and Library ABIE definitions and each of their constituent ASBIE and BBIE members as arrays, not on the declarations of the content of the ASBIE and BBIE objects that define the members of the arrays. The “title” annotation shall be CCTS Dictionary Entry Name for the BIE. The “description” annotation shall be the CCTS definition value.

6.5 JSON Schema fragments and declarations

6.5.1 JSON schema fragments for Document ABIEs

FRG21 Document ABIE JSON schema fragments

There shall be one JSON schema fragment created for each Document ABIE, declared to be a “JSON draft v4 schema” with the appropriate “\$schema” string property defined as “<http://json-schema.org/draft-04/schema#>” in addition to the annotation title and description for the fragment.

FRG22 Document ABIE object namespace declarations

Each Document ABIE JSON schema fragment shall include a root schema object declaration including declarations for four properties as string values, all optional, those being for namespaces of record providing for transliteration with XML instances if desired. The string values shall be named:

- “_D” for the Document ABIE namespace
- “_S” for the Library ABIE namespace used for ASBIEs
- “_B” for the BBIE namespace
- “_E” for the extension scaffolding namespace

FRG23 Document ABIE object reference declaration

The Document ABIE JSON schema fragment root schema object declaration shall include a single required property declaration for a array of maximum and minimum one item, named for the Document ABIE, referencing its definition locally in the file under the “definitions” object.

Note

While the prescribed referencing provides no additional facility for a Document ABIE when compared to an inline object definition, this declaration pattern mimics the declaration pattern that is required for Library ABIEs, and thus is required here simply for consistency. Generation tools may find this a convenience.

FRG24 Document ABIE object definition declaration

Each Document ABIE schema fragment shall include a “definitions” object that contains a single object definition declaration, that being for the content of the Document ABIE.

6.5.2 JSON schema fragment for Library ABIEs

FRG25 Library ABIE JSON schema fragment

There shall be one common schema fragment created to contain all ASBIEs (that is, from every Document ABIE and every Library ABIE) and all Library ABIEs. This schema fragment shall not have a “\$schema” property and shall include only a “definitions” object in addition to the annotation title and description for the fragment.

FRG25 Library ABIE JSON schema fragment

Example

In UBL 2.2 the <code>UBL-CommonAggregateComponents-2.2.json</code> fragment serves this purpose.

FRG26 Library ABIE object reference declarations

The one Library ABIE schema fragment shall include in its “definitions” object an object reference declaration for every ASBIE in the model (that is, from every Document ABIE and every Library ABIE) that is not named the same as its corresponding Library ABIE. Each such declaration shall reference its corresponding Library ABIE without any title or description properties.

FRG27 Library ABIE object definition declarations

The one Library ABIE schema fragment shall include in its “definitions” object an object definition declaration for every Library ABIE, each being for its content.

There are no constraints on the order of the ABIE declarations in the schema expression.

Note

This lack of an order constraint may seem in conflict with the semantic model library constraint of alphabetical order of ABIE components. Whereas the semantic ABIE components are organized for the benefit of the human reader, the order of the schema component declarations does not affect the schema processor. However, using alphabetical order in the schema fragment may be a convenience for the human reader of the schema expressions.

6.5.3 JSON schema fragment for BBIEs

FRG28 BBIE JSON schema fragment

There shall be one common schema fragment created to describe all BBIEs in the model (that is, from every Document ABIE and every Library ABIE). This schema fragment shall not have a “\$schema” property and shall include only a “definitions” object in addition to the annotation title and description for the fragment.

Example

In UBL 2.2 the <code>UBL-CommonBasicComponents-2.2.json</code> fragment serves this purpose.

FRG29 BBIE object definition declarations

The one BBIE schema fragment shall include a “definitions” object that contains an object definition declaration for every BBIE in the model (that is, from every Document ABIE and every Library ABIE). Each such declaration shall reference its corresponding qualified or unqualified data type without any title or description information.

There are no constraints on the order of the BBIE declarations in the schema expression.

Note

The order of the schema component declarations does not affect the schema processor. However, using alphabetical order in the schema fragment may be a convenience for the human reader of the schema expressions.

6.5.4 JSON schema declarations for ABIEs

DCL21 ABIE object declaration

Every ABIE shall be declared as an object named by the CCTS Component Name of the ABIE. It shall have as its title the CCTS Dictionary Entry Name. It shall have as its description the CCTS Definition. It shall declare as the “required” property the names of the ASBIE and BBIE children whose model cardinality has a minimum bound of 1. Its properties shall be the ASBIE and BBIE declarations of the ABIE’s children. It shall declare that additional properties are not permitted.

Example

```
"CatalogueRequestLine": {  
  "title": "Catalogue Request Line. Details",  
  "description": "A class to define a line describing a  
  request for a catalogue line.",  
  "required": [  
    "ID",  
    "Item"  
  ],  
  "properties": {...},  
  "additionalProperties": false,  
  "type": "object"  
},
```

DCL22 Library ABIE object declaration content order

The members of a Library ABIE shall be ordered as the sequence (in the order the BIEs appear in the semantic model of the ABIE) of all BBIE children references first, followed by all ASBIE children references.

Example

```
"CatalogueRequestLine": {  
  "title": "Catalogue Request Line. Details",  
  "description": "A class to define a line describing a  
  request for a catalogue line.",  
  "required": [  
    "ID",  
    "Item"  
  ],  
  "properties": {  
    "ID": {..."items": {  
      "$ref":  
"UBL-CommonBasicComponents-2.2.json#/definitions/ID"  
    }},...},  
    "ContractSubdivision": {..."items": {  
      "$ref":  
"UBL-CommonBasicComponents-2.2.json#/definitions/  
ContractSubdivision"  
    }},...},  
    "Note": {..."items": {  
      "$ref":  
"UBL-CommonBasicComponents-2.2.json#/definitions/Note"  
    }},...},  
    "LineValidityPeriod": {..."items": {
```

DCL22 Library ABIE object declaration content order

```
        "$ref": "#/definitions/Period"
      }, ... },
      "RequiredItemLocationQuantity": { ... "items": {
        "$ref": "#/definitions/ItemLocationQuantity"
      }, ... },
      "Item": { ... "items": {
        "$ref": "#/definitions/Item"
      }, ... }
    },
    "additionalProperties": false,
    "type": "object"
  },
```

Note

Although the order of properties of a JSON object is not relevant, ordering the properties as prescribed is consistent with both the CCTS model and the XML serialization of the model. For the human reader of the JSON schema, having this order in the schema declarations should be helpful.

DCL23 Document ABIE object declaration content order

The members of a Document ABIE shall be ordered first with a reference to the extension collection object array, followed by the sequence (in the order the BIEs appear in the semantic model of the ABIE) of all BBIE children references first, followed by all ASBIE children references. The extension collection array shall have a minimum and maximum number of items of 1 and shall not allow additional properties. The array shall not be listed in the "required" property.

Example

```
"ApplicationResponse": {
  "title": "Application Response. Details",
  "description": "A document to indicate the
application's response to a transaction. This may be a
business response initiated by a user or a technical
response sent automatically by an application.",
  "required": [
    "ID",
    "IssueDate",
    "SenderParty",
    "ReceiverParty"
  ],
  "properties": {

    "UBLExtensions": {
      "title": "UBLExtensions",
      "description":
"An optional set of extensions to the committee model",
      "items": {
        "$ref":
"../common/UBL-CommonExtensionComponents-2.2.json#/
definitions/UBLExtensions"
      },
      "maxItems": 1,
      "minItems": 1,
```

DCL23 Document ABIE object declaration content order

```
"additionalProperties": false,
"type": "array"
},

"UBLVersionID": {...},
"CustomizationID": {...},
"ProfileID": {...},
"ProfileExecutionID": {...},
"ID": {...},
"UUID": {...},
"IssueDate": {...},
"IssueTime": {...},
"ResponseDate": {...},
"ResponseTime": {...},
"Note": {...},
"VersionID": {...},
"Signature": {...},
"SenderParty": {...},
"ReceiverParty": {...},
"DocumentResponse": {...}
},
"additionalProperties": false,
"type": "object"
}
```

Note

Although the order of properties of a JSON object is not relevant, ordering the properties as prescribed is consistent with both the CCTS model and the XML serialization of the model. For the human reader of the JSON schema, having this order in the schema declarations should be helpful.

6.5.5 JSON schema declarations for ASBIEs

DCL24 ASBIE property declaration in an ABIE object

Every ASBIE child of an ABIE shall be declared as an array named by the CCTS Component Name of the ASBIE. It shall have as its title the CCTS Dictionary Entry Name. It shall have as its description the CCTS Definition. It shall have a minimum number of items as 1. If the cardinality has a maximum bound of 1, then the declaration shall have a maximum number of items as 1, otherwise there shall be no constraint on the maximum number of items. It shall declare that additional properties are not permitted. The items of the array shall be declared by referencing the ASBIE in the Library ABIE schema fragment.

Example

```
"SenderParty": {
  "title": "Application Response. Sender_ Party. Party",
  "description": "The party sending this document.",
  "items": {
    "$ref":
    "../common/UBL-CommonAggregateComponents-2.2.json#/
    definitions/SenderParty"
  },
  "maxItems": 1,
```

DCL24 ASBIE property declaration in an ABIE object

```
"minItems": 1,  
"additionalProperties": false  
"type": "array",  
},
```

DCL25 ASBIE declaration in the Library ABIE JSON schema fragment

In the Library ABIE schema fragment, every ASBIE child of an ABIE shall be declared by referencing the ASBIE's associated ABIE within the same fragment. There shall be no title, description or other properties.

Example

```
"SenderParty": {  
  "$ref": "#/definitions/Party"  
},
```

6.5.6 JSON schema declarations for BBIEs

DCL26 BBIE property declaration in an ABIE object

Every BBIE child of an ABIE shall be declared as an array named by the CCTS Component Name of the BBIE. It shall have as its title the CCTS Dictionary Entry Name. It shall have as its description the CCTS Definition. It shall have a minimum number of items as 1. If the cardinality has a maximum bound of 1, then the declaration shall have a maximum number of items as 1, otherwise there shall be no constraint on the maximum number of items. It shall declare that additional properties are not permitted. The items of the array shall be declared by referencing the BBIE declaration in the BBIE schema fragment using the CCTS Component Name of the BBIE.

Example

```
"ResponseDate": {  
  "title": "Application Response. Response Date. Date",  
  "description": "The date on which the information in  
the response was created.",  
  "items": {  
    "$ref":  
    "../common/UBL-CommonBasicComponents-2.2.json#/  
definitions/ResponseDate"  
  },  
  "maxItems": 1,  
  "minItems": 1,  
  "additionalProperties": false  
  "type": "array",  
},
```

DCL27 BBIE declaration in the BBIE JSON schema fragment

In the BBIE schema fragment, every BBIE shall be declared by referencing the BBIE's type's declaration in either the qualified data type fragment or the unqualified data type fragment as required. There shall be no title, description or other properties.

DCL27 BBIE declaration in the BBIE JSON schema fragment

Example of a BBIE declaration with a qualified data type

```
"SourceCurrencyCode": {  
  "$ref":  
  "UBL-QualifiedDataTypes-2.2.json#/definitions/  
  Currency_CodeType"  
},
```

Example of a BBIE declaration with an unqualified data type

```
"SourceCurrencyBaseRate": {  
  "$ref":  
  "UBL-UnqualifiedDataTypes-2.2.json#/definitions/  
  RateType"  
},
```

6.5.7 JSON schema declarations for Qualified Data Types

DCL28 Qualified data type declaration in the qualified data type JSON schema fragment

Every qualified data type shall be declared with its name being the type's Dictionary Entry Name compressed with all periods and spaces removed. It shall have as its title the type's Dictionary Entry Name. It shall reference the associated unqualified data type in the unqualified data type schema fragment. There shall be no description or other properties.

Example

```
"Currency_CodeType": {  
  "title": "Currency_ Code. Type",  
  "$ref":  
  "UBL-UnqualifiedDataTypes-2.2.json#/definitions/  
  CodeType"  
},
```

6.6 Extension JSON schema fragments and declarations

6.6.1 Extension information in JSON

The content type for a Document ABIE contains a single optional extension collection object defined as an array of exactly one extension property in order to provide for the inclusion of data in the JSON expression that is in addition to the data of the information bundle for the document. Such data may include content designed by other organizations as well as augmentations of the semantic model.

The extension property is declared as an array of one or more extension objects. Optionally, each extension object has a suite of metadata properties used to describe the extension. Each extension object must have a required extension content array of exactly one extension content object. The extension metadata and content may reuse existing ABIEs or BBIEs and may contain JSON content not modeled as BIEs.

The extension scaffolding and metadata are JSON objects modeled using JSON schema fragments and constructs independent of the schema fragments created for the semantic model.

6.6.2 Extension collection JSON schema fragments and declarations

EXT21 Extension collection JSON schema fragment

There shall be one common extension collection JSON schema fragment created to include the declarations of the extension collection object, the extension object, the extension content element, the extension metadata objects and any required type information for metadata objects that are not BIEs in the Library ABIE schema fragment. The extension schema fragment shall not have a "\$schema" property and shall include only a "definitions" object in addition to the annotation title and description for the fragment.

Example

In UBL 2.2 the UBL-CommonExtensionComponents-2.2.xsd fragment serves this purpose.

EXT22 Extension content object declaration

The extension collection JSON schema fragment shall include the declaration of the mandatory extension content object, but not the definition information for the extension content object.

Note

The rationale for not including the definition information for the extension point object is that the type is subject to change, while the extension collection, the extension item and extension item metadata object and type information all are not. This separation allows the extension collection and item JSON schema fragment to be deployed as read-only, while the extension point data type JSON schema fragment can be deployed as writable in order to be overridden with a definition created by users.

EXT23 Extension collection object content

The document's extension collection shall be declared as having an array property of one or more extension objects as its content. The array property shall have a minimum number of items of "1". It may have a "description" property. It shall declare that additional properties are not permitted. The array items shall reference the extension object definition in the same schema fragment.

Example

```
"UBLExtensions": {
  "description": "A container for all extensions present
in the document.",
  "required": [
    "UBLExtension"
  ],
  "properties": {
    "UBLExtension": {
      "description":
        "A single extension for private
use.",
      "items": {
```

EXT23 Extension collection object content

```
        "$ref": "#/definitions/UBLExtension"
      },
      "minItems": 1,
      "type": "array",
      "additionalProperties": false
    }
  },
  "additionalProperties": false,
  "type": "object"
},
```

Note

The rationale for providing for multiple extensions is that different users of a document may have different extension items added to the content. Also, different extensions may be thematically distinguished (e.g. the digital signature extension is semantically separate from an extension augmenting invoice line content).

EXT24 Extension object content ordering

The extension object shall declare all available metadata objects (if any) in advance of a single extension content property listed last. The extension content property shall be listed in the “required” property as well as any extension metadata that may be mandatory. There are no constraints on the available properties describing extension metadata, but there shall be a declaration of no additional properties.

Example

```
"UBLExtension": {
  "description": "A single extension for private use.",
  "required": [
    "ExtensionContent"
  ],
  "properties": {
    "ID": {...},
    "Name": {...},
    "ExtensionAgencyID": {...},
    "ExtensionAgencyName": {...},
    "ExtensionVersionID": {...},
    "ExtensionAgencyURI": {...},
    "ExtensionURI": {...},
    "ExtensionReasonCode": {...},
    "ExtensionReason": {...},
    "ExtensionContent": {...}
  },
  "additionalProperties": false,
  "type": "object"
}
```

Note

There are no constraints on the selection, name, definition or cardinality of the extension metadata elements.

EXT25 Extension object property content declarations

The content property and each metadata property of the extension object shall be declared as an array. Each property shall declare a minimum number of items as "1". The property for the content shall declare a maximum number of items as "1". The array for the content item shall reference the extension content data type JSON schema fragment. The properties for the metadata items with a maximum cardinality of "1" shall declare a maximum number of items as "1". The array for each metadata property shall reference a definition in one of the Library ABIE, BBIE, Qualified Data Type or Unqualified Data Type JSON schema fragments.

Example

```
"ID": {
  "description": "An identifier for the Extension
assigned by the creator of the extension.",
  "items": {
    "$ref":
"UBL-CommonBasicComponents-2.2.json#/definitions/ID"
  },
  "maxItems": 1,
  "minItems": 1,
  "type": "array",
  "additionalProperties": false
},
"ExtensionReason": {
  "description": "A description of the reason for the
Extension.",
  "items": {
    "$ref":
"UBL-UnqualifiedDataTypes-2.2.json#/definitions/
TextType"
  },
  "maxItems": 1,
  "minItems": 1,
  "type": "array",
  "additionalProperties": false
},
"ExtensionContent": {
  "description": "The definition of the extension
content.",
  "items": {
    "$ref":
"UBL-ExtensionContentDataType-2.2.json#/definitions/
ExtensionContent"
  },
  "maxItems": 1,
  "minItems": 1,
  "type": "array",
  "additionalProperties": false
}
},
```

Note

There are no constraints on the selection, name, definition or maximum cardinality of the extension metadata properties.

6.6.3 JSON schema fragment for the extension content data type declaration

EXT26 Extension content data type JSON schema fragment

There shall be one extension content data type schema fragment created to include the declaration of the content type for the extension content object. This schema fragment shall not have a “\$schema” property and shall include only a “definitions” object in addition to the annotation title and description for the fragment.

Example

In UBL 2.2 the UBL-ExtensionContentDataType-2.2.xsd fragment serves this purpose.

EXT27 Extension content data type declaration

The extension content element type schema fragment shall contain only a single object declaration comprised of any content without constraint.

Example

```
"ExtensionContent": {  
  "description":  
    "A user-defined repository of additional content",  
  "type": "object"  
}
```

Note

The rationale for the lax validation is to allow for the extension point to contain, without error, any information that is supplemental to the business document but not defined by the semantic model.

6.7 Qualified data types JSON schema fragment and declarations

QDT21 Qualified data types JSON schema fragment

There shall be one qualified data types schema fragment created to include the declarations of any qualified data types referenced in the schema fragment for BBIEs. This schema fragment shall not have a “\$schema” property and shall include only a “definitions” object in addition to the annotation title and description for the fragment.

Example

In UBL 2.2 the UBL-QualifiedDataTypes-2.2.json fragment serves this purpose.

QDT22 Qualified data type JSON declaration name

Every qualified data type shall be declared using the Dictionary Entry Name name of the data type compressed by removing periods and spaces. The title property of the declaration shall be the uncompressed Dictionary Entry Name.

QDT23 Qualified data type JSON declaration basis

Every qualified data type shall be based on an unqualified data type, and may impose as additional constraints whatever qualifications are required to be expressed using JSON schema semantics.

Note

In UBL 2.2 there are no qualifications expressed using JSON schema semantics. Every qualified data type declaration simply makes reference to its associated unqualified data type declaration.

QDT24 Qualified data type JSON declaration constraint

Every qualified data type declaration shall be such that every possible instance of the declared type is also an instance of the base type.

Note

This constraint prevents additions of anything that is not part of the base unqualified data type, such as the introduction of any new properties, or any less-constrained property values than the constraints on the unqualified data type.

6.8 Unqualified data types JSON schema fragment and declarations

UDT21 Unqualified data types JSON schema fragment

There shall be one unqualified data types schema fragment created to include the declarations of all unqualified data types referenced in the schema fragment for BBIEs. This schema fragment shall not have a "\$schema" property and shall include only a "definitions" object in addition to the annotation title and description for the fragment.

Example

In UBL 2.2 the `UBL-UnqualifiedDataTypes-2.2.json` fragment serves this purpose.

UDT22 Unqualified data types declaration inclusions

An unqualified data type shall be declared for every one of the permitted Primary Representation Terms and the Secondary Representation Terms defined as Permissible Representation Terms in the Core Component Technical Specification [CCTS] Section 8-3 "Permissible Representation Terms".

Note

It may be a convenience to implementers to generate this JSON schema fragment by transforming an XSD expression of all possible unqualified data types as any particular CCTS model for documents may not be comprehensively using all possible unqualified data types. For example, the UBL 2.2 unqualified data type JSON schema fragment was generated from the `UBL-Unqualified-2.2.xsd` schema fragment

UDT23 Unqualified data types declaration exclusions

Unqualified data types shall only be declared for the permitted Primary Representation Terms and the Secondary Representation Terms defined as Permissible Representation Terms in the Core Component Technical Specification [CCTS] Section 8-3 “Permissible Representation Terms”.

UDT24 Unqualified data types declaration base

Every unqualified data type shall be either a reference to one of the approved Core Component Types defined in the Core Component Technical Specification [CCTS] JSON schema fragment, or be an independent object declaration of constraints that satisfies the intent of the type. The “title” property shall be the Dictionary Entry Name and the “description” shall include the definition. An independent declaration shall have a set of properties based on the content and supplementary components of the type. The name of object properties shall be prefixed with the CCTS Dictionary Entry Name compressed by removing the word “Type” from the end and by removing periods and spaces. The name of the content property shall be suffixed with the word “Content”. The names of the supplemental component properties shall be suffixed with the CCTS Dictionary Entry Name of the supplementary component compressed by removing periods and spaces. The data types of the subordinate declarations shall be numbers or strings as appropriate to the corresponding XSD declared type. The object shall declare that additional properties are not permitted.

Note

The rational for having UDT declarations is to impose some JSON syntax semantics on top of the more general decimal and string lexical syntax defined in the CCTS specification of Core Component Types. For example, the CCTS Core Component Type for date and time is a simple string without constraint. Such lax structuring of the field value does not serve users in that no particular format is obligated. The UDT can impose, for example, the JSON date primitive type lexical syntax on all date and time values, overriding the CCTS definition.

Note

This rule does allow an optional supplementary component defined in the CCTS Core Component Type not to be available in the associated unqualified data type. For example, if the UDT implements a built-in JSON primitive type for the component then there is no use of a format supplementary component and associated attribute and so the format attribute declaration can be omitted and, thereby, be unavailable for use for that data type.

Example 1

In this example the unqualified data type “Value. Type” uses the Core Component Type “Numeric. Type” without change:

```
"ValueType": {
  "title": "Value. Type",
  "description": "Numeric information that is assigned
or is determined by calculation, counting, or
sequencing. It does not require a unit of quantity or
unit of measure.",
  "$ref":
"CCTS_CCT_SchemaModule-2.2.json#/definitions/
NumericType"
},
```

UDT24 Unqualified data types declaration base

Example 2

In this example the unqualified data type “Date Time. Type” replaces the Core Component Type “Date Time. Type” with no attributes and with a JSON built-in primitive type for content:

```
"DateTimeType": {
  "title": "Date Time. Type",
  "description": "A particular point in the progression
of time, together with relevant supplementary
information.",
  "properties": {
    "DateTimeContent": {
      "type": "string",
      "format": "date-time"
    },
    "additionalProperties": false,
    "type": "object"
  }
},
```

UDT25 Unqualified data types declaration constraint

Every unqualified data type declaration that is not a reference to one of the Core Component Type declarations defined in the Core Component Technical Specification [CCTS] JSON schema fragment shall be such that every possible instance of the declared type is also an instance of one of the Core Component Types as defined in CCTS.

Note

This constraint prevents additions of anything that is not part of the base Core Component Type, such as the introduction of any new properties, or any less-constrained property values.

Example

In this example the unqualified data type for a date is based on the JSON primitive type for date-time, with the additional constraint of not including a time value, and thus all instances of the date data type are also instances of the string-based DateTimeType data type in CCTS:

```
"DateType": {
  "title": "Date. Type",
  "description": "One calendar day according the
Gregorian calendar.",
  "properties": {
    "DateContent": {
      "allof": [
        {
          "type": "string",
          "format": "date-time"
        },
        {
          "type": "string",
```

UDT25 Unqualified data types declaration constraint

```
        "pattern": "^-*[^T:-]+-[^T:]+$"
      }
    ],
    "additionalProperties": false,
    "type": "object"
  }
},
```

6.9 CCTS Core Component Types JSON schema

All data types in the Core Component Technical Specification [CCTS] JSON schema fragment correspond to the 10 CCTS Primary Representation Terms defined in [CCTS] Section 8-3 “Permissible Representation Terms”.

CCT21 CCTS Core Component Type JSON schema fragment

There shall be one core component type schema fragment of primitive data types for primary representation terms on which the unqualified data types are based. This schema fragment shall not have a “\$schema” property and shall include only a “definitions” object in addition to the annotation title and description for the fragment.

This schema fragment’s definitions shall be derived from the XSD schema fragment published by UN/CEFACT with the following embedded title and metadata:

CCTS Core Component Type Schema Module

Module of Core Component Type

Agency: UN/CEFACT

VersionID: 1.1

Last change: 14 January 2005

Example

In UBL 2.2 the CCTS_CCT_SchemaModule-2.2.xsd fragment serves this purpose.

CCT22 Core Component Type property declarations

Every core component type shall be declared as an object named by the Dictionary Entry Name compressed by removing periods and spaces. It shall have as its title the CCTS Dictionary Entry Name. It shall have as its description the CCTS Definition. It shall have as its properties the content declaration as well as a declaration of each of its supplemental components. The name of these properties shall be prefixed with the CCTS Dictionary Entry Name compressed by removing the word “Type” from the end and by removing periods and spaces. The name of the content property shall be suffixed with the word “Content”. The names of the supplemental component properties shall be suffixed with the CCTS Dictionary Entry Name of the supplementary component compressed by removing periods and spaces. The data types of the subordinate declarations shall be numbers or strings as appropriate to the corresponding XSD declared type. The object shall declare that additional properties are not permitted.

Example

```
"MeasureType": {  
  "title": "Measure. Type",  
  "description": "A numeric value determined by  
measuring an object along with the specified unit of  
measure.",  
  "properties": {  
    "MeasureContent": {  
      "type": "number"  
    },  
    "MeasureUnitCode": {  
      "type": "string"  
    },  
    "MeasureUnitCodeListVersionIdentifier": {  
      "type": "string"  
    }  
  },  
  "additionalProperties": false,  
  "type": "object"  
},
```

7 Conformance

7.1 An information bundle and its associated validation artefacts conforming to these naming and design rules do not violate any rule or requirement expressed in normative sections of this specification related to modeling (clauses 3 and 4) and one's choices of validation artefacts. The XSD and CVA artefacts (clause 5) are for documents expressed in XML syntax according to the model. The JSON schema artefacts (clause 6) are for documents expressed in JSON syntax according to the model.

- 7.2 [ATT01 Leading name part in attribute names \(5.10\)](#)
- 7.3 [ATT02 Non-leading abbreviations in attribute names \(5.10\)](#)
- 7.4 [CCT01 CCTS Core Component Type schema \(5.9\)](#)
- 7.5 [CCT21 CCTS Core Component Type JSON schema fragment \(6.9\)](#)
- 7.6 [CCT22 Core Component Type property declarations \(6.9\)](#)
- 7.7 [COM01 Dictionary information values \(4.2\)](#)
- 7.8 [COM02 Dictionary information value prohibited characters and character sequences \(4.2\)](#)
- 7.9 [COM03 Controlled list of abbreviations in BIE names \(4.3\)](#)
- 7.10 [COM04 Controlled list of abbreviations in dictionary entry name information values \(4.3\)](#)
- 7.11 [COM05 List of equivalent terms in BIE names \(4.3\)](#)
- 7.12 [COM06 Component Type for a BIE \(4.4.1\)](#)
- 7.13 [COM07 Minimum set of dictionary information values describing an ABIE \(4.4.2\)](#)
- 7.14 [COM08 Minimum set of dictionary information values describing a BBIE \(4.4.3\)](#)
- 7.15 [COM09 Minimum set of dictionary information values describing an ASBIE \(4.4.4\)](#)
- 7.16 [COM10 Dictionary entry name uniqueness \(4.4.5\)](#)
- 7.17 [COM11 CCTS dictionary information item name value prohibited characters \(4.4.5\)](#)
- 7.18 [COM12 Use of leading upper case letter in dictionary entry name values \(4.4.5\)](#)
- 7.19 [COM13 ABIE contents cannot be empty \(4.5\)](#)
- 7.20 [COM14 ABIE children ordering \(4.5\)](#)
- 7.21 [DCL01 Element declarations \(5.5.4\)](#)
- 7.22 [DCL02 Element declaration references \(5.5.4\)](#)
- 7.23 [DCL03 Type declarations \(5.5.4\)](#)
- 7.24 [DCL04 ABIE element declaration \(5.5.5\)](#)
- 7.25 [DCL05 ABIE type declaration \(5.5.5\)](#)
- 7.26 [DCL06 Library ABIE type declaration content order \(5.5.5\)](#)
- 7.27 [DCL07 Document ABIE type declaration content order \(5.5.5\)](#)

- 7.28 [DCL08 Document ABIE extension element cardinality \(5.5.5\)](#)
- 7.29 [DCL09 ASBIE schema element declaration \(5.5.6\)](#)
- 7.30 [DCL10 BBIE element declaration \(5.5.7\)](#)
- 7.31 [DCL11 BBIE unqualified type declaration \(5.5.7\)](#)
- 7.32 [DCL12 BBIE qualified type declaration \(5.5.7\)](#)
- 7.33 [DCL13 Qualified data type declaration \(5.5.8\)](#)
- 7.34 [DCL21 ABIE object declaration \(6.5.4\)](#)
- 7.35 [DCL22 Library ABIE object declaration content order \(6.5.4\)](#)
- 7.36 [DCL23 Document ABIE object declaration content order \(6.5.4\)](#)
- 7.37 [DCL24 ASBIE property declaration in an ABIE object \(6.5.5\)](#)
- 7.38 [DCL25 ASBIE declaration in the Library ABIE JSON schema fragment \(6.5.5\)](#)
- 7.39 [DCL26 BBIE property declaration in an ABIE object \(6.5.6\)](#)
- 7.40 [DCL27 BBIE declaration in the BBIE JSON schema fragment \(6.5.6\)](#)
- 7.41 [DCL28 Qualified data type declaration in the qualified data type JSON schema fragment \(6.5.7\)](#)
- 7.42 [DTQ01 Data type qualification CVA file \(5.11\)](#)
- 7.43 [DTQ02 Data type element content qualifications \(5.11\)](#)
- 7.44 [DTQ03 Data type attribute content qualifications \(5.11\)](#)
- 7.45 [DTQ04 Value test constraints \(5.11\)](#)
- 7.46 [DTQ05 Value list constraints \(5.11\)](#)
- 7.47 [DTQ06 Value metadata association \(5.11\)](#)
- 7.48 [EXT01 Extension collection schema fragment \(5.6.2\)](#)
- 7.49 [EXT02 Extension content element declaration \(5.6.2\)](#)
- 7.50 [EXT03 Extension collection element content \(5.6.2\)](#)
- 7.51 [EXT04 Extension element content ordering \(5.6.2\)](#)
- 7.52 [EXT05 Extension content data type schema fragment \(5.6.3\)](#)
- 7.53 [EXT06 Extension content data type declaration \(5.6.3\)](#)
- 7.54 [EXT07 Extension content data type imports \(5.6.3\)](#)
- 7.55 [EXT21 Extension collection JSON schema fragment \(6.6.2\)](#)
- 7.56 [EXT22 Extension content object declaration \(6.6.2\)](#)
- 7.57 [EXT23 Extension collection object content \(6.6.2\)](#)
- 7.58 [EXT24 Extension object content ordering \(6.6.2\)](#)
- 7.59 [EXT25 Extension object property content declarations \(6.6.2\)](#)

- 7.60 [EXT26 Extension content data type JSON schema fragment \(6.6.3\)](#)
- 7.61 [EXT27 Extension content data type declaration \(6.6.3\)](#)
- 7.62 [FRG01 Document ABIE schema fragments \(5.5.1\)](#)
- 7.63 [FRG02 Document ABIE element declaration \(5.5.1\)](#)
- 7.64 [FRG03 Document ABIE type declaration \(5.5.1\)](#)
- 7.65 [FRG04 Library ABIE schema fragment \(5.5.2\)](#)
- 7.66 [FRG05 Library ABIE element declarations \(5.5.2\)](#)
- 7.67 [FRG06 Library ABIE type declarations \(5.5.2\)](#)
- 7.68 [FRG07 BBIE schema fragment \(5.5.3\)](#)
- 7.69 [FRG08 BBIE element declarations \(5.5.3\)](#)
- 7.70 [FRG09 Library ABIE type declarations \(5.5.3\)](#)
- 7.71 [FRG21 Document ABIE JSON schema fragments \(6.5.1\)](#)
- 7.72 [FRG22 Document ABIE object namespace declarations \(6.5.1\)](#)
- 7.73 [FRG23 Document ABIE object reference declaration \(6.5.1\)](#)
- 7.74 [FRG24 Document ABIE object definition declaration \(6.5.1\)](#)
- 7.75 [FRG25 Library ABIE JSON schema fragment \(6.5.2\)](#)
- 7.76 [FRG26 Library ABIE object reference declarations \(6.5.2\)](#)
- 7.77 [FRG27 Library ABIE object definition declarations \(6.5.2\)](#)
- 7.78 [FRG28 BBIE JSON schema fragment \(6.5.3\)](#)
- 7.79 [FRG29 BBIE object definition declarations \(6.5.3\)](#)
- 7.80 [MOD01 Document ABIE \(3.1\)](#)
- 7.81 [MOD02 ABIE library contents \(3.2\)](#)
- 7.82 [MOD03 ABIE library ordering \(3.2\)](#)
- 7.83 [MOD04 Extension availability \(3.3\)](#)
- 7.84 [NAM01 Namespaces for information found in information bundles \(5.2\)](#)
- 7.85 [NAM02 Namespaces for an extension \(5.2\)](#)
- 7.86 [NAM03 Namespaces for BBIE data types \(5.2\)](#)
- 7.87 [NAM21 Namespaces for information found in information bundles \(6.2\)](#)
- 7.88 [QDT01 Qualified data types schema fragment \(5.7\)](#)
- 7.89 [QDT02 Qualified data type declaration name \(5.7\)](#)
- 7.90 [QDT03 Qualified data type declaration basis \(5.7\)](#)
- 7.91 [QDT04 Qualified data type declaration constraint \(5.7\)](#)

- 7.92 QDT21 Qualified data types JSON schema fragment (6.7)
- 7.93 QDT22 Qualified data type JSON declaration name (6.7)
- 7.94 QDT23 Qualified data type JSON declaration basis (6.7)
- 7.95 QDT24 Qualified data type JSON declaration constraint (6.7)
- 7.96 UDT01 Unqualified data types schema fragment (5.8)
- 7.97 UDT02 Unqualified data types declaration inclusions (5.8)
- 7.98 UDT03 Unqualified data types declaration exclusions (5.8)
- 7.99 UDT04 Unqualified data types declaration base (5.8)
- 7.100 UDT05 Unqualified data types declaration constraint (5.8)
- 7.101 UDT21 Unqualified data types JSON schema fragment (6.8)
- 7.102 UDT22 Unqualified data types declaration inclusions (6.8)
- 7.103 UDT23 Unqualified data types declaration exclusions (6.8)
- 7.104 UDT24 Unqualified data types declaration base (6.8)
- 7.105 UDT25 Unqualified data types declaration constraint (6.8)

Appendix A (informative) Release notes

A.1 Availability

Online and downloadable versions of this release are available from the locations specified at the top of this document.

A.2 Status of this release

Release of this package initiates the process to develop Committee Specification 01 of version 1.1.

A.3 Package structure

This Committee Specification Draft 01 / Public Review Draft 01 is published as a zip archive in the <http://docs.oasis-open.org/ubl/Business-Document-NDR/v1.1/csprd01/> directory. Unzipping this archive creates a directory tree containing a number of files and subdirectories. Note that while the two XML files comprise the revisable version of this specification, this revisable XML may not be directly viewable in all currently available web browsers.

The base directory has the following files:

Business-Document-NDR-v1.1-csprd01.xml

The revisable form of the document.

Business-Document-NDR-v1.1-csprd01-summary.xml

A distillation of the rules, comprising only the rule title and the section number in which the rule is found. The title is linked to the rule and the section number is linked to the section. This XML document is incorporated in the revisable form of the document by way of an entity reference. During the publishing process this file is first distilled from the revisable form and then subsequently incorporated in the revisable form for publishing.

Business-Document-NDR-v1.1-csprd01.html

An HTML rendering of the document.

Business-Document-NDR-v1.1-csprd01.pdf

A PDF rendering of the document.

These are the subdirectories in the package:

art

Diagrams and illustrations used in this specification

db

DocBook stylesheets for viewing in HTML the XML of this work product

A.4 Release history

Version 1.0 of the Business Document Naming and Design Rules describes the constraints on the CCTS modeling [CCTS] of XML interchange documents [XML] and the creation of corresponding W3C schema [XSD schema] and OASIS Context/Value Association documents [CVA].

Version 1.1 of the Business Document Naming and Design Rules adds constraints on the creation of CCTS corresponding JSON interchange expressions [ISO 21778 - ECMA JSON] using JSON Schema expressions [JSON Schema]. In Version 1.1 there are no additions, removals or changes in the rules

for CCTS modeling or for XML interchange documents compared to Version 1.0. Sections 1 and 2 are modified slightly to incorporate JSON in the scope of work. Sections 3, 4 and 5 are unchanged save for disambiguating subsection titles from the newly-introduced subsections. Section 6 is added with the information on JSON schema. Section 7 replaces the former section 6 on conformance and it incorporates the rules related to JSON.

Appendix B (informative) Acknowledgements

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Todd Albers, Federal Reserve Bank of Minneapolis
Oriol Bausa Peris, Individual
Kenneth Bengtsson, Individual
Erlend Klakegg Bergheim, Difi-Agency for Public Management and eGovernment
Kees Duvekot, RFS Holland Holding B.V.
G. Ken Holman, Crane Softwrights Ltd.
Ole Madsen, Danish Agency for Digitisation, Ministry of Finance
Enric Staromiejski Torregrosa, everis, S.L.U.

Appendix C (informative) Additional production processes

C.1 CCTS serialization

An implementation of these naming and design rules may choose to create a serialization of the CCTS information. This can be a useful convenience when processing the CCTS information as a whole. The CCTS collaboration tool is not required to produce a serialization if such is not needed to fulfill its obligation to produce validation artefacts.

There are no formal rules for CCTS serialization.

C.2 Reporting

An implementation of these naming and design rules is not required to produce any supplementary reports. These reports can be useful reference materials for review of the CCTS information maintained for the information bundles.

There are no formal rules for reporting.

Appendix D Temporary Annex - Change Log (Non-Normative)

Note

This temporary appendix will be removed in the final version of the committee specification.

Revision	Date		Edit- or	Changes made
csd01wd01	04 2017	January	GKH	Initial version of 1.1
csprd01	11 2017	January	GKH	Cover page changes for public review; copy-fitting code samples